Representative Techniques for Siting and Environmental Analysis of Road Corridors

William O. Rasmussen

University of Arizona

September 1979

Final Report

for

Cooperative Agreement Nos. 16-594-GR and 16-703-GR

Eisenhower Consortium For Western Environmental Forestry Research

Rocky Mountain Forest and Range Experiment Station

LIBRARY

NOV 16 1979

ROCKY MOUNTAIN STAT

Representative Techniques for Siting and Environmental Analysis of Road Corridors

William O. Rasmussen

University of Arizona

September 1979

Final Report

for

Cooperative Agreement Nos. 16-594-GR and 16-703-GR

Eisenhower Consortium For Western Environmental Forestry Research

Rocky Mountain Forest and Range Experiment Station

Representative Techniques for Siting and Environmental Analysis of Road Corridors

Preface

Introduction

Suitability Surface

Path Location

Manual Path Definition

Dynamic Programming Path Definition

Characteristics of PATH Defined Paths

Directional Biasing Path Definition

- Possible Movements
- Cell Geometry Corrections
- Path Development

Characteristics of TRAIL Defined Paths

Example Path Selection Using PATH, TRAIL, and TRACK

Path Expansion to a Corridor

Site Specific Environmental Analysis

Summary

Acknowledgements

Literature Cited

Appendix A - Computer Program SLMATH

Appendix B - Computer Program TRACK

Appendix C - Computer Program PATH

Appendix D - Computer Program TRAIL

PREFACE

This is the second report describing the various aspects of a representative sample of computer software for use by a transportation planner as tools in the delineation of recreational road corridors in a forest environment. Various environmental and social considerations are taken into account in the production of a road corridor suitability surfaces for the area.

This report deals with the problem of choosing or delineating a path through such a matrix of road suitability values. Three techniques are examined along with their particular biasing. Additionally there is a discussion of several environmental simulators which may be used to assess the environmental effects of converting the land use to that of a road corridor.

Introduction

Selection of the "best" location of a road corridor across a suitability surface may be considered a two step analysis. First, data relative to the area, are assembled and a suitability surface defining location suitability for the feature is developed. This suitability surface may be thought of as reflecting relative costs, both economic and environmental, of positioning the road at one location relative to another on the study area. The second step in the road corridor location analysis involves the delineation of the road corridor centerline on the suitability surface of the area. Several approaches exist for location of a path through a suitability data array, however, most of these approaches may be placed into one of three major technique classes. This report discusses these three techniques for locating a linear feature on a suitability surface, and presents some of their associated biases.

Once a corridor route has been selected, to link two or more points together, the environmental consequences of the corridor alignment may be examined on a site-by-site basis using other more detailed, computer tools (Larsen et al. 1978). Several of these are outlined herein.

Suitability Surface

The suitability data array is the product of mathematical operations applied to the digital form of the individual cellular geographic resource data sets for the area (Rasmussen 1978). All data used in the computation of the suitability surface are geographic in nature, that is they have

geographic coordinates associated with them. These coordinates are used by appropriate computer programs to allow the data to be interacted and manipulated with other data topics location by location (cell by cell) throughout the area.

The conversion of a map to a digital form is achieved through the use of an electronic point digitizer. This device produces a coordinate pair of map coordinates for any selected point on a map. In the case of topography, the elevation of the point whose coordinates were calculated by the digitizer is input to the computer as the third number of a data triplet. The computer program SYMAP (Dougenik and Sheehan, 1975) is used to interpolate from the set of all input data points characterizing a topographic map to produce average elevation values for each cell location in a rectangular array of cells used to represent the map area. Each cell has the same area. The resultant data array, from the interpolation of the input topographic map data, is one topic of information which is used with others in the computation of the suitability surface.

In the case of a map of fixed value sub areas, such as a soils map, the boundaries of the soil types are approximated by a series of points using the digitizer. All cells in the final array lying predominantly within the polygon boundary are assigned a value representative of the enclosed soil type. This operation is repeated for all soil type boundaries on the soils map. Most maps fall into one of these two classes of map format, i.e., either point values such as elevation or constant value sub areas such as soils maps. Other maps, such as timber density, soil erodibility, esthetics, etc. are input and stored in a geographics data base. Such data may conveniently be displayed and manipulated with the program SLMATH (Appendix A).

One technique for the definition and assemblage of a suitability surface for location of a recreational road through an area has been previously

covered (Rasmussen 1978). In that report the assumption was made that the suitability surface represented a mix of economic costs, for construction of the road, and esthetic costs, in terms of the visual impact to the person driving along the road. The road suitability surface is computed for the case of limited funds and no consideration of the visual impact resulting in the road suitability data array being the same as the economic data array. For the other extreme the suitability surface is the same as the esthetic surface of the area. In general the suitability surface is some mix of these two. The delineation of the "best" road location on such a suitability surface is one which goes through those cells having the lowest local values and also, when added, give the lowest total road cost.

Path Location

Path selection techniques usually operate on either the selection of the path with the highest values associated with it (using suitability data) or through cells with the lowest values (cost or sensitivity data). For purposes of discussion the path through the data array will be located such as to pass through cells with low numerical values relative to others in the array. A cost surface is used in such a case. The path will be located such that the total path value (sum of all cells passed through) will be minimal and the average value for a single path cell will be low.

The location of the path, which when widened defines the road corridor through a suitability surface, has been achieved using three techniques. As mentioned, many of the procedures or conceptual schemes for location of a path through a suitability data array, presented in the literature, fall into one of the three techniques presented herein.

The three techniques consist of 1) a manual positioning approach where a person may interactively position a path on a cost surface placed on a digitizer, 2) a dynamic programming algorithm which determines the global optimum path that is, the lowest of all possible total costs, and 3) a local neighborhood search algorithm which uses directional biasing to search for the "best" path through the data array.

Manual Path Definition

The manual location of a path on a cost surface is a convenient means of introducing complex constraints regarding the placement of the path. The path may be located such that it either passes through or stays away from any specified locations. The location constraints on a road might be such that it should go near certain visitor facilities in a forest but stay away from certain animal habitats. These locations, for either attempting to go near or avoid, are flagged or marked on the cost surface across which the road is to be located. The location of the road in areas other than in the neighborhood of these items is guided by placement of the road through cells which have low costs for road location and at the same time such that the total length of the road is minimal to help keep total costs down.

Once a cost surface for road location has been developed, using as many considerations as appropriate, a manual location of the road may be achieved using a printer map representation of the cost surface as a guide in the delineation of the road. The printer map of the cost surface is placed on a digitizer bed and the location of the road is drawn on the map. The line drawn on the printer map is then digitized along with registration points to allow computation of the cells in the cost data array which are

to be traversed by the road. A computer program, TRACK (Appendix B), calculates the coordinates of all cells which will have the road passing through them. A masking data array is then developed where values of 1.0 are used for all cell locations where the digitized road passes through while all other cells are assigned a value of 0.0. The multiplication of this road mask array by an array of cost data values gives as a product array non zero values only to those cells which have the road passing through them. All other cells have a value of 0.0 assigned to them. The sum of all cost values along the road is computed in addition to the average cost value per road cell. Other road locations may be delineated on the map with the computer program computing the total and average costs for comparison among paths. The road location mask may also be used on other data topics to determine the total and average values of those parameters along the path selected for the road.

The manual interactive placement of the path requires the operator to keep several relations in mind. The total path value for a long path going only through cells with very low values could easily be greater than another path which contains fewer cells of higher value. The location of a path only through cells with low values must be tempered with the selection of a path that is as short as possible. The optimum path is selected such that it passes through cells which have the lowest values possible and the total length is a minimum. Thus, it may be advantageous to have high values for a part of the path if the total path value will be reduced. The manual placement of the path allows it to be positioned in locations where tunneling or passing through areas of high values is an appropriate procedure. This may allow the overall or total path to have a lower value than other paths which do not pass through areas with high data values. For operations

of this nature, manual delineation of a path is superior to the directional bias approach (see TRAIL discussion).

In the manual placement of a path on the cost surface the operator's mind selects locations which have the lowest local values but can not adequately sum up the cells to determine if the total value of the path is lower than some other. For selection of the lowest total value path, the computer is very beneficial as it can determine the lowest total cost path from several possibilities with ease. Interactive and sometimes iterative operation of TRACK helped to determine the lowest total path cost from several possibilities.

Dynamic Programming Path Definition

The second path selection process involves the use of a dynamic programming algorithm to find the global optimum in terms of the path with the minimal value for the total path cost. The path cost is arrived at by the addition of all cell values traversed by the path. The algorithm is based on the work of Dijkstra (1959) and later modified by others (Elsner et al., 1975; Giles et al., 1976). This algorithm with further modifications has been developed into the computer program PATH (Appendix C).

The path selected across an area using the program often corresponds closely to the straight line connecting the two terminal points. Deviations from a straight line may be indicated by variability in the data. Requirements on the data, in the cost data array, for existence on an optimum path deviating from a straight line between two terminal points are not totally obvious. The optimum path is defined as one where the sum of all path element (cell) values is a minimum relative to the totals of all other possible paths.

For a constant value cost surface with N cells separating the terminal points, the minimum total path value, T, is a straight line. If we assume a sequential notation of cells on a path the total value is given by

$$T = \sum_{i=1}^{N} a_i \quad . \tag{1}$$

where a_i is the value of cell i in the path

In the case of a constant surface, all cell values are equal to the same constant value, c, such that

$$T = \sum_{i=1}^{N} c = nc \quad . \tag{2}$$

All cell values, a_{ij}, in a suitability data array with variability are observed to have a maximum cell value, a_H, for the entire array given by

$$a_H = \max \{a_{ij}\} \text{ for all } (i,j) \quad . \tag{3}$$

For such a data array the largest total value for a straight line path between the terminal points would be given by

$$T_H = \sum_{i=1}^{N} a_H = Na_H \quad . \tag{4}$$

This assumes all members of the path to have values equal to the greatest value in the data array.

The minimum cell value, a_L, for the data array set of values a_{ij} is given by

$$a_L = \min \{a_{ij}\}, \text{ for all } (i, j). \tag{5}$$

The lowest total value for a straight line path, T_L, between the terminal points would be composed of all cells having a value equal to the lowest value in the data array, a_L, where

$$T_L = \sum_{i=1}^{N} a_L = Na_L . \tag{6}$$

The total value of the optimum path, T_o, between the two terminal points is bounded by

$$T_L \leq T_o \leq T_H ,$$

$$Na_L \leq T_o \leq NA_H ,$$

$$a_L \leq \frac{T_o}{N} \leq a_H . \tag{7}$$

If we assume the optimum path is M cells long, where M is greater than N then the total value, T_o, is given by

$$T_o = \sum_{i=1}^{M} a_i . \tag{8}$$

The value of each cell may be represented by a deviation, b_i, from the average cell value, c, for the entire data array such that each cell

in the path, a_i, may be represented by

$$a_i = b_i + c \ . \qquad (9)$$

Thus, for a path of M cells to be better or lower than the maximum value of N cells the relationship of

$$\sum_{i=1}^{M} a_i \leq \sum_{i=1}^{N} a_H \ , \qquad \text{or}$$

$$\sum_{i=1}^{M} (b_i+C) \leq Na_H \ , \qquad \text{or}$$

$$\sum_{i=1}^{M} b_i + MC \leq Na_H \ , \qquad \text{or}$$

$$\sum_{i=1}^{M} b_i \leq Na_H - MC \qquad \text{holds.} \qquad (10)$$

For a path of M cells to have a total value equal to a path of N cells inplies

$$\sum_{i=1}^{M} a_i = \sum_{i=1}^{N} a_i \ , \qquad \text{or}$$

$$\sum_{i=1}^{M} (b_i+C) = \sum_{i=1}^{N} (b_i+C) \ , \qquad \text{or}$$

$$\sum_{i=1}^{M} b_i + MC = \sum_{i=1}^{N} b_i + NC \ , \text{ or}$$

$$(M-N)C = \sum_{i=1}^{N} b_i - \sum_{i=1}^{M} b_i \quad . \tag{11}$$

If the value of N is 10 and M is 11 then equation (11) requires the difference in the sums of deviations from the array average of path values in the two paths must be equal to the average cell value c. For the optimum path to deviate greatly from a straight line would generally require large deviations in the data if the deviations are randomly distributed. Conversely, from a knowledge of the data variability the potential deviation from a straight line of the optimum path is implied. Straight lines are optimum paths on constant surfaces.

Assuming the cell values are randomly distributed on two data arrays, and the variability of average change between adjacent cells is the same for one array and much greater for the other, then equation (11) implies there may be a longer optimum path length on the data array with the lower average cell value. This is only a potential condition; the actual distribution of values in the array relative to the terminal points defines what is possible. However, in general, a data array with a lower average cell value, and same data variability as another array often may have a longer optimum path associated with it. The addition of a large value to all cells in an array tends to force the optimum path to a straight line solution.

Equation (11) implies the lengthening of a path by one cell requires the sum of the deviations from the array average to equal one average cell value. If the average cell value is high, it may easily be this condition cannot be met. As the average cell value is lowered, with the same variability, this condition becomes easier to fulfill. Thus, by appropriate changes in the average cell values of an array, the optimum path may be shortened or lengthened.

Characteristics of PATH Defined Paths

The location of the path through the use of the dynamic programming algorithm is such that data variability may modify it. The transformation of the cost surface, or reciprocal suitability surface, by the subtraction of a constant value from all cell values may cause the minimum path to be positioned in a different manner and longer than before the operation was performed. Similarily, the addition of a constant value will tend to cause the minimum path to shorten and approach a straight line solution. The higher the individual cell values and the less the value change of adjacent cells, the more probable the optimum path will be aligned closely with a straight line connecting the terminal points.

One means of lengthening the path would be if the cost array is first operated on by the subtraction of the lowest value found in the array for the study area from all cell values and the resulting data array linearily transformed such that the data lies in the range .01 to 10.0. The location of the minimal path on this surface will likely follow more closely the lows on the surface and will probably be longer in total length than the path on the original surface.

The addition of a value to each cell in the data array will tend to force the path, using the dynamic programming algorithm, to be a straight line. The extension of this concept to one limit would be the assignment of an enormous value to each cell. Then the sum of adjacent cell value fluctuations across the data array, from terminal point A to B, would be less than the value of a single cell. The path location in this case would be the one which minimizes the number of cells in the path connecting the terminal points, i.e., a straight line. A similar constraint is applied to

inter-cell variability. If the entire data array were made up of the same value, no matter how large or small, then the optimum path with the lowest total value would again be the path representing a straight line. Data variability as well as average value are parameters which imply the potential path deviation from a straight line.

Directional Biasing Path Definition

The computer program TRAIL (Appendix D) searches for a path through a cost data array by examination of cells in the immediate neighborhood, using directional biasing to determine the best move for least cost and also movement towards a goal cell (link or terminal point). TRAIL is input two terminal points, along with any intermediate points which are to be passed through. Three-directional bias variables, to be discussed later, are also input. TRAIL then computes coordinates of all cells in the determined path and outputs these in the form of a table and a printer map.

Possible Movements

Cell a_7 (figure 1) may be thought of as the current head of a path being computed through the data array. The path was incremented from position a_6. In such a case, movement to a_6 is not allowed as the path may not back up. In addition, cells to which movement may be made are further reduced by allowing only forward movement. This allows movement to be made next to one of e_2, e_3, or e_4.

Directional biasing is achieved by adding weighting values, W_i, to cells e_2, e_3, and e_4. The magnitude of the value added is a function of

how much the path is to be constrained from going in a given direction. The addition of a large value tends to keep the path from going in that direction. The next cell in the path is the one of the three forward movement cells which has the lowest total value, $(e_i + w_i)$ (figure 2). This selected cell will be called path cell a_8. The procedure is then repeated with cell a_8 in place of a_7.

Cell Geometry Corrections

Individual cells are rectangular, with the distance between cell centers of adjacent rows, and the same column, given by dr (figure 3). The distance between cell centers of adjacent columns and the same row is represented by dc. The diagonal distance, d, between cell a, and e_4 of figure 1 is given by

$$d = \sqrt{(dc)^2 + (dr)^2} . \tag{12}$$

In the current situations the cell sides have the ratio of 8:10 for de:dr such that

$$\frac{dc}{dr} = \frac{8}{10} ,$$

$$dc = .8dr,$$

and

$$dr = 1.25dc. \tag{13}$$

This implies

$$d = \sqrt{.64dr^2 + dr^2},$$

$$= 1.28\ dr, \tag{14}$$

or in terms of dc

$$d = 1.60dc. \tag{15}$$

Numerical values in cells e_i (figure 4) are adjusted for variations in the distance between centers using the relationships:

$$\text{distance } (a_7 - e_3) = dc,$$

$$\text{and} \quad \text{distance } (a_7 - e_2) = \text{distance } (a_7 - e_4) = 1.60dc. \tag{16}$$

The values in cell e_3 remains the same. The values in cells e_2 and e_4 are modified to be

$$e_2' = 1.60\ e_2, \tag{17}$$

$$\text{and} \quad e_4' = 1.60\ e_4. \tag{18}$$

The cells for decision as to best next location for designation as an element of the path are, after being adjusted for distance between cell centers, represented in figure 5. These cell values have been multiplied by distance weights between a_7 and their cell centers in units of dc (the distance between adjacent cells on a given row).

Path Development

For a temporary end of the path, c, at cell (i,j), the geometric relationship between that cell and the end cell, D, of the present path link is represented in figure 6.

Directional bias weights are assigned cells such that a full weight is added to the cell which contains the line segment $\overline{CD}$ while the cell opposite the direction of D is given a weight 0.0 with intermediate cells given a weight which is a linear function of the angle between the direction of the final cell, D, and the neighboring cell center.

The weight W is a function of 3 weights, w_1, w_2, and w_3,

$$W = f_1 + f_2 + f_3 , \tag{19}$$

where

$$f_1 = \frac{w_1}{(\text{Distance from end cell B})} ,$$

$$f_2 = w_2 ,$$

and $f_3 = w_3$ * (Distance along path from start at A). These weights are input by the user at the start of the program.

The total path between terminal points may be broken into several smaller segments or links. For each segment, temporary start and end points are determined. The program operates as if these were the overall end points for the path. Path information before or after this link is not used in the path positioning for the current link. The sum of all path segments or links is the total path from A to B in figure 7.

The link end points are calculated from input of the minimum and maximum forward search distance to be used to locate the link end point from a given starting point.

For the end point of the first link the location of the line between terminal points A and B is computed in terms of which cells would be traversed to define a straight line. A search band is defined with boundaries perpendicular to $\overline{AB}$. The near, L min, and far, L max, boundaries to terminal point A are fixed distances input by the operator.

All cells in the search band have a value computed for the perpendicular distance from the cell center to $\overline{AB}$. A weight, W_4, is input whose value is such that larger values of it tend to force the end link point to lie close to $\overline{AB}$. A weighting value, W_{ij} is now added to each cell value which is a function of the distance, D_{ij}, of cell (i,j) from $\overline{AB}$ and the constant weight W_4,

$$W'_{ij} = d_{ij} + W_4 \quad , \quad \text{for all } (i,j) \text{ contained in the search band.} \tag{20}$$

These weights are applied to all cells in the search band to give

$$e'_{ij} = e_{ij} + w'_{ij} \tag{21}$$

$$= e_{ij} + d_{ij} + w_4, \quad \text{for all } (i,j)' \text{ contained in the search band.}$$

The cell with the lowest value in the search band is defined as a swath through the data array perpendicular to $\overline{A_1B}$, where A_1 is the end of the first link.

Distances of all cell centers in search band 2 from $\overline{A_1B}$ are computed

and the value of each cell is modified by

$$e'_{ij} = w'_{ij} + e_{ij} \qquad (22)$$

$$= d_{ij} + w_4 + e_{ij}, \quad \text{for all } (i, j) \text{ contained in search band 2.}$$

The cell having the lowest value in the search band is defined as the end point of the second link (A_2). A path is next computed between A_1 and A_2. At the completion of this a third link end point is sought in search band 3. This is continued until the search band contains the final terminal point B, in this case the terminal point is defined as the final link end point. The total path is then the sum of the links

$$[A,B] = [A,A_1] + [A_1,A_2] + \ldots + [A_N,B], \qquad (23)$$

where

$[A_m, A_{m+1}]$ represents the path or link from A_m to A_{m+1}.

Characteristics of TRAIL Defined Paths

This program will compute paths which either meander and follow the local lows in the cost surface when directional weights are low or approximate a straight line when the weights are set very high. The setting of high weights causes the path computed to closely approximate that found by the dynamic programming technique (PATH). What is achieved then is the selection of a path which gets to the end point fast and passes through as many lows along the way as possible. This is often near the optimal path linking the end points determined from PATH.

The setting of the weights very low allows the path to lie in local lows throughout its length. It attempts to follow valleys or depressions from one terminal point to the other.

It is quite informative to better comprehend the nature of the surface to run TRAIL with low, medium, and high directional weights to see the paths selected in each mode and then compute the total and average cost for each path. The effect of path length on cost is quite easily seen from such an exercise for a specific cost surface.

Linear transformations in the cost data do not appreciably affect the location of the path across it when using TRAIL. Relative to PATH the program TRAIL computes paths which are quite insensitive to a constant uniform increase or reduction in the cost data array.

Example Path Selection Using PATH, TRAIL, AND TRACK

Several data sets were developed for Woods Canyon in the Coconino National Forest of Arizona. These were used to produce an array of values depicting the cost of road construction per mile for each cell location into which the watershed had been divided. The cost of road construction was a function of several factors such as slope, size of the overstory, nature of the soil, and type of subsurface material (Rasmussen 1978). Another generated array was the scenic beauty values for the area where the range in scenic beauty had been transformed to lie in the same numeric range as the construction cost data.

These two data arrays had various paths sited on them. For paths computed across the economic surface the only consideration was construction cost while for paths across the scenic beauty surface the only

consideration was scenic beauty or esthetics. In the first case the road was located to minimize construction cost while in the later the road was placed through areas which people find pleasing to look at.

The two programs, PATH and TRAIL, were used to develop paths across the two data arrays.

For the case of the economic map PATH was used to compute path 1 across the surface (figure 9). The range of the economic data was then reduced from the range of 8,310-33,218 to that of 0-24,908, and another operation of PATH produced path 5 for the best location of the line connecting the two selected end points such that the total path cost was a minimum.

Program TRAIL was then used with tight, medium and loose directional weights on the 8,310-33,218 economic data to produce paths 6, 7, and 8 as the best paths across the surface (figure 9).

For the case of the scenic beauty data array PATH selected path 4 as the best connector of the end points while with loose direction weights TRAIL selected path 3 and then with tight direction weights it selected path 2 (figure 10).

These eight paths are shown positioned on both the economic surface and the scenic beauty surface (figures 9 and 10). Program TRACK was then used to compute the total cost and average cell cost for all paths on the economic surface with a similar computation of all paths on the scenic beauty surface. These values are presented in table 1. Such a table is used to ascertain the construction cost for a road positioned using only scenic beauty criteria and the scenic beauty cost for a road positioned using only economic criteria. Various mixes of costs could also be determined as discussed in a previous report (Rasmussen 1978).

By observing the relative positioning of the various paths one becomes aware of the effect of changing the weights in TRAIL on the selected path and the effect of changing the data range in PATH for computation of the best path. For certain cases one or the other of these programs might be used. Such comparisons allows more knowledgeable decisions to be made as to which is best to use in a given situation. Other programs might also be examined before a specific one selected for a given project (Turner et al., 1970; Giles et al., 1976; Elsner et al., 1975; Miller et al., 1972; Weisz et al., 1975).

As can be seen TRACK is well suited for computing the cost of a path computed on one surface in terms of values from another surface while it was primarily designed for a manual delineation of a path across a surface. Paths selected by TRACK are mainly a function of the skill of the user and thus were not presented for comparison with the paths from TRAIL and PATH.

Another tool useful for comprehension of path location on a topographic (or other base) is perspective plotting (figure 11). The computer program used here was called SYMVU (Rens, 1971).

Path Expansion to a Corridor

Once the path has been selected by the appropriate process or computer program, a corridor may be developed by widening it.

The corridor is converted into a binary mask used to reduce the area of analysis in the geographic data base in the final alignment process to only cells within the corridor.

As an example, the corridor data array stored in the computer has a value of 1.0 assigned to all cells contained in the corridor with a value

of 0.0 given to all cells outside. Multiplication of the corridor data array, cell-by-cell, with a topographic array results in only topographic cells in the corridor having non-zero values. The topographic data array was thus reduced such that only cells in the corridor have values. This resultant elevation array is used as input data to OPTLOC (Bennett, 1973) or FSRDS (George, 1975) for calculating the final and optimum alignment of a road connecting the terminal points lying within the corridor. A similar operation may be performed on each data topic to present only data sets for the area within the proposed coridor. Often it is advisable to have these data sets further analyzed using ecosystem component simulation models at specific sites along the corridor. Such an analysis, if it is performed, should occur before the final corridor is passed to OPTLOC or FSRDS as it's alignment may be adjusted due to predicted adverse effects from the environmental simulators.

Site Specific Environmental Analysis

Since the multi-topic suitability analysis uses only rough indicators of the environmental consequences of corridor location in a given area, site specific ecosystem component simulation models may be used to yield more detailed predictions of the response of the ecosystem when subjected to corridor management. The environmental consequences of the decision to change the existing land use to that of a road corridor may be assessed, as may the consequences of various management practices.

A family of interactive modular components that facilitate decision-making with respect to land use planning in the southwestern United States

are currently under development (Larsen et al.,1978). These models are well suited for site specific corridor evaluation. While progress in designing, refining, and verifying continues, an outline of the conceptual framework of this computer system will illustrate this general approach to ecosystem simulation and use in corridor evaluation.

As structured at this time, the family of modular components consists of: a set of general modules that, in turn, include simulators designed to estimate specific ecosystem responses to management activities; an interactive command system through which users link selected modular components and retrieve source data to operate those modules; and associated data bases to be introduced into a simulation exercise as required. To allow continual updating, necessary in meeting anticipated user demands, maximum flexibility is insured within the framework of the system.

To illustrate application of the family of modular components, a description of the general modules which estimate specific ecosystem responses to management activities is presented. Particular emphasis is placed on components which simulate the ecosystem responses for conversion to a road corridor land use. As the complete array of modular components cannot be described in this paper, only the simulation of herbaceous understory, animal carrying capacities, habitat quality for a variety of animal species, water yield, and sedimentation will be highlighted.

In the first case, an interactive modular component has been developed to estimate production and (to some extent) composition of herbaceous understories from knowledge of competing shrub overstories, precipitation amounts, and if appropriate, time since the implementation of a management activity which reduces the impact of competing shrub overstories. While initial efforts have been directed toward the estimation of total herbaceous understories,

subsequent additions to the component will allow the simulation and partitioning of herbage production into three general categories: perennial grasses and grass-like plants, forbs and half-shrubs, and shrubs. Such partitioning will provide better insight to land use planning for anticipated consumption of the herbaceous resource by domestic livestock and wildlife species. This component is used to predict the effect on the herbaceous understory to reduction of the overstory in the corridor. The information obtained may then be used to compute animal carrying capacity of the corridor in the event that is of interest.

In a modular component structured to determine animal carrying capacities, the production of herbaceous understories (entered directly by the user or obtained through outputs generated by the herbage production simulator) is analyzed with respect to usable forage components for cattle, sheep, goats, and wildlife species. Appropriate plant species for inclusion in each forage component were ascertained from existing literature relevant to the preferred foods for these animals. Proper use factors are introduced into the simulation exercise by the user in an attempt to meet specific management objectives. Similarly, the amount of usable forage required to support the animal species of concern is generally input by the user to achieve a desired end. Given the time which animals will actually be consuming forage on the tract of land under consideration, along with an identification of possible constraints which may affect the distribution of animals, carrying capacities are determined through simulation and then compared to actual stocking rates to assess land use in terms of animal grazing activities.

Assessment of wildlife habitat quality for a variety of game, nongame, and domestic animals is achieved by another modular component. This component involves the synthesis of wildlife habitat quality prior to and following the

implementation of management practices which in turn, are summarized and arrayed in a pattern recognition model. Within this component, animals are assigned numerical values ranging from 0 to 10, with habitat quality for a particular animal species in a given ecosystem increasing with numerical value. Specific assignment of these values is achieved through analysis of mathematical functions that relate habitat preference to expressions of vegetative density.

As originally developed, habitats for most animals being considered have been conceptualized as two components: available food and cover (protective and thermal). Mathematical functions describing changes in food and cover for particular animal species as related to the spatial distributions of forest overstories are defined from existing information. These food and cover functions are then integrated to provide a single function that reflects the changes in these habitat components with respect to the spatial distributions of forest overstories.

By comparing habitat quality values that represent conditions prior to a management change with those predicted for habitats modified by management redirection, either an increase (+), a decrease (-), or no change (0) is determined. Then, a matrix, of pluses, minuses, and zero arrayed for all animal habitats, and management alternatives of interest (a pattern recognition model) can be displayed to provide insight into comparative management impacts.

Another modular component has been formulated to estimate water yield from readily available inputs (i.e., precipitation amounts, soil characteristics, and densities of shrub overstories). Outputs from the model include daily water yield, change in soil moisture, evapotranspiration, and interception.

Coupled with the water budget simulator is a component which predicts the amount of suspended sediment in streamflow. This sediment component, which requires knowledge of the spatial distribution of organic material and the streamflow regime as inputs, can estimate maximum concentrations and total weights of sediment under different land management activities.

Another model is one for predicting erosion from a new road. The model uses as input a description of the road template and the nature of the soil and vegetative cover on exposed surfaces. The model predicts the erosion rate for various segments of the road cross section and the amount of erosion per mile of right-of-way. The erosion rate varies as vegetative cover becomes established or other changes occur in the surface cover.

Many of the inputs for certain models may be outputs from other models. Default data are offered the user if he does not have a full data set. These default data are average or typical values for the specific ecosystem being simulated. These values are overridden by the user if he has data better suited for his specific simulation.

To provide simulation opportunities for other natural resource considerations, modular components have also been synthesized to predict: growth and yield of forest overstories; development and accumulation of organic material on the ground; maximum concentrations and daily volumes of dissolved chemical constituents in streamflow; and asethetic values associated with different ecosystems and man's activities. Collectively, all of the components can be used to describe the impacts of converting from the existing land use to that of a road corridor. They may also be used to predict impacts of various corridor management practices on an existing corridor.

If through operation of the various modular components one finds a location within the corridor to exhibit an unacceptable response, the original

corridor analysis can be repeated with the location of the sensitive area noted on one of the factors used in the analysis such that the specific location is removed, for all practical purposes, from consideration of corridor use. The new corridor is then evaluated with the site specific modular components to see if there are any problems. If problems exist, the process is repeated.

SUMMARY

Many of the existing computer programs for location of a road corridor through an area represented by a corridor location suitability surface may be divided into one of three classes. First, there are techniques which rely on manual placement of the corridor through the area, second there are dynamic programming and related techniques, and thirdly there are search algorithms with directional weights for determining the best path across a road corridor suitability surface.

One program representing each of these three techniques is presented herein and compared against the other two. In addition there are listings of the computer program's FORTRAN IV instruction sets in the appendicies. Attached to these listings are instructions on how to operate each program.

In order to have suitability surfaces, geographic resource data sets may be used with the program SLMATH to generate them as outlined in a previous report (Rasmussen 1978). This program and instructions for its use are also contained in an appendix.

Best paths were computed across an economic and a scenic beauty surface representing Woods Canyon watershed in central Arizona. Various parameters

within the path selection programs were varied to show the variations which might be expected in location of the predicted best path across the area. A linear reduction in the economic data set was seen to cause a variation in the location of the best path across the surface using TRAIL.

By use of the program TRACK, optimum road corridor paths computed across the economic surface were similarly placed across the scenic beauty surface for comparison of the total cost of the path in economic and scenic beauty units. Conversely optimum paths computed across the scenic beauty surface were placed on the economic surface for comparison of total costs of the path in both units.

Once the best path across the suitability surface was located it may be widened to produce a corridor across the area. The land within this corridor is then examined in more detail relative to its suitability for use as a road corridor. Such analysis is aided through the use of numerous environmental simulators. These are site specific but may be operated several times to cover the length of the corridor. In the event these simulators predict sections of the corridor to be unsuitable for such use, this information may be folded into the main suitability analysis and the entire operation repeated to determine another corridor which goes around the unsuitable area.

The programs contained within this report whould make the transportation planner aware of the various techniques currently available to assist in location of a suitable road corridor through an area based on environmental, social, and economic criteria. He might also go through the literature cited here and other places to see if there are systems well suited for his own needs. A fair number of these computer tools exist. He should look around before choosing an inadequate tool or trying to reinvent the

wheel. Hopefully this and previous reports (Rasmussen, 1978; Weisz et al., 1975) will be of some help to the planner.

ACKNOWLEDGEMENTS

The author wishes to thank Alma Sperr, Herbert Schroeder and John Burns for their able computer programmings, and Peter Ffolliott for his input with the environmental simulators. The project was carried out at the University of Arizona utilizing their DEC-10 and CYBER 175 computers. It was funded, in part, by the Eisenhower Consortium for Western Environmental Forestry Research, the Rocky Mountain Forest and Range Experiment Station, and Western Regional Research W-125.

LITERATURE CITED

Bennett, David W. 1973. OPTLOC II General Description and User's Manual. University of Melbourne, Australia, June 1973. 124 pp.

Dijkstra, E.W. 1959. A note on two problems in connection with graphs. Numerical Mathematics Vol. 1, pp. 269-271.

Dougenik, J.A., and D.E. Sheehan. 1975. SYMAP USER'S REFERENCE MANUAL. Labortory for Computer Graphics and Spatial Analysis, Harvard Univ., Cambridge, MA, 178 pp.

Elsner, G.H., M.R. Travis, and P.H. Kourts. 1975. DYNAMIC PROGRAMMING SUBROUTINES BASED ON THE DIJKSTRA ALGORITHM FOR FINDING MINIMUM COST PATHS IN DIRECTED NETWORKS. Information Report FF-X-51, Forest Fire Research Institute, Ottawa, Ontario, 16 pp.

George, Thomas A. 1975. "The Forest Service's Computer-Aided Road Design System", Transportation Research Board Special Report 160, pp. 75-81.

Giles, Robert H., A. Blair Jones III, and Charles W. Smart. 1976. POWER: A COMPUTER SYSTEM FOR CORRIDOR LOCATION, Office of Biological Services, Fish and Wildlife Service, U.S. Dept. of the Interior, Washington, DC, 30 pp.

Larson, F.R., P.F. Ffolliott, W.O. Rasmussen and D.R. Carder. 1979. ESTIMATING IMPACTS OF SILVICULTURAL MANAGEMENT PRACTICES ON FOREST ECOSYSTEMS. In Best Management Practices for Agriculture and Silviculture, Ann Arbor Science Publishers Inc., Ann Arbor, MI, pp. 281-294.

Miller, Allen H., and Bernard J. Niemann, Jr. 1972. An Interstate Corridor Selection Process: The Application of Computer Technology to Highway Location Dynamics--Phase I. Environmental Awareness Center, Dept. of Landscape Architecture, Univ. of Wisconsin/Madison.

Rasmussen, W.O. 1978. DEVELOPMENT OF A CORRIDOR SELECTION SYSTEM FOR FOREST ROAD LOCATION PLANNING. Final Report, Eisenhower Consortium Cooperative Agreement No. 16-526-CA, Rocky Mtn. Forest and Range Exp. Station, Ft. Collins, CO, 28 pp.

Rens, F.J. 1971. SYMVU MANUAL. Laboratory for Computer Graphics and Spatial Analysis, Harvard University, Cambridge, MA, 60 pp.

Turner, A.K., and R.D. Miles. 1970. THE GCARS SYSTEM: A COMPUTER-ASSISTED METHOD OF REGIONAL ROUTE LOCATION. Highway Research Record, # 348, pp. 1-15.

Weisz, Reuben N., D. A. King, and Ralph Kingery. 1975. Planning for Forest Roads - - State of the Art in Corridor Selection, Report on Eisenhower Consortium Cooperative Agreement 16-447-CA, Rocky Mtn. Forest and Range Exp. Station, Ft. Collins, CO, 100 pp.

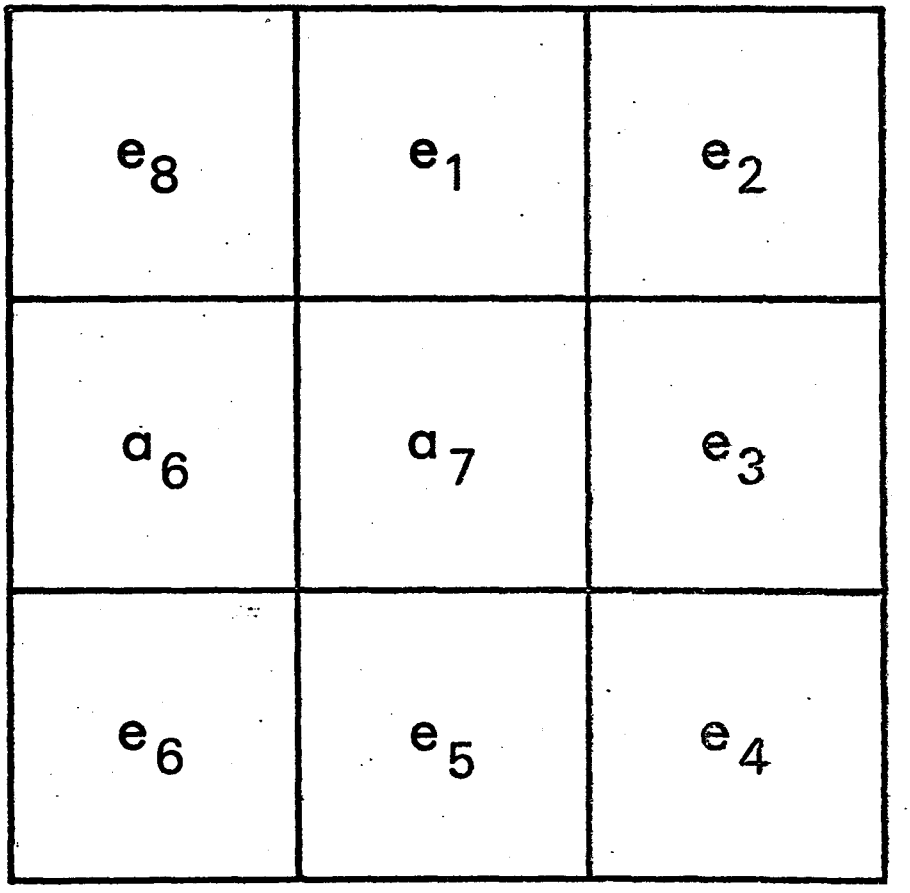

Figure 1 - Cells available for movement from end of path cell, a_7.

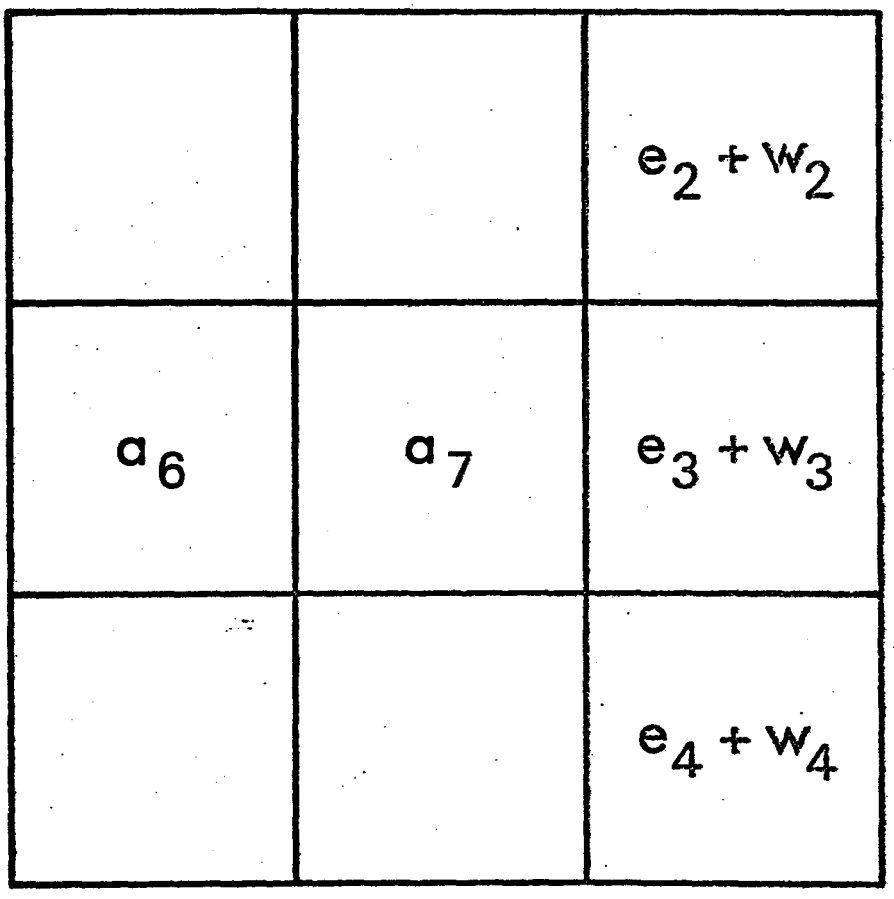

Figure 2 - Addition of weight to those cells where movement is possible.

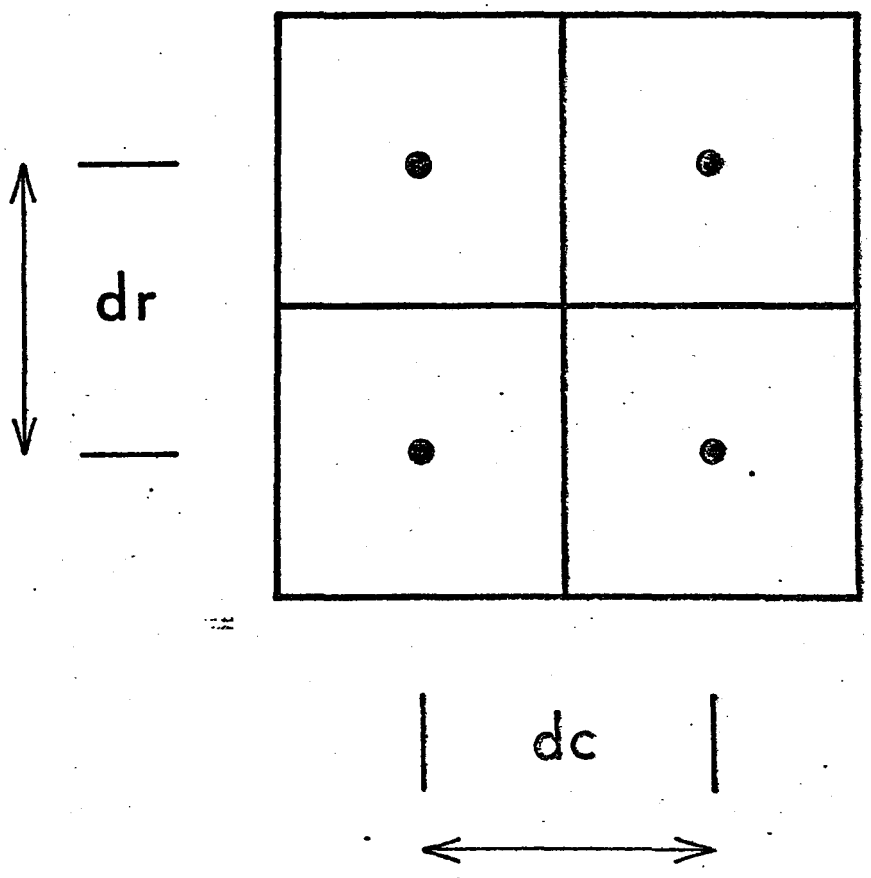

Figure 3 - Column and row distance between cell centers.

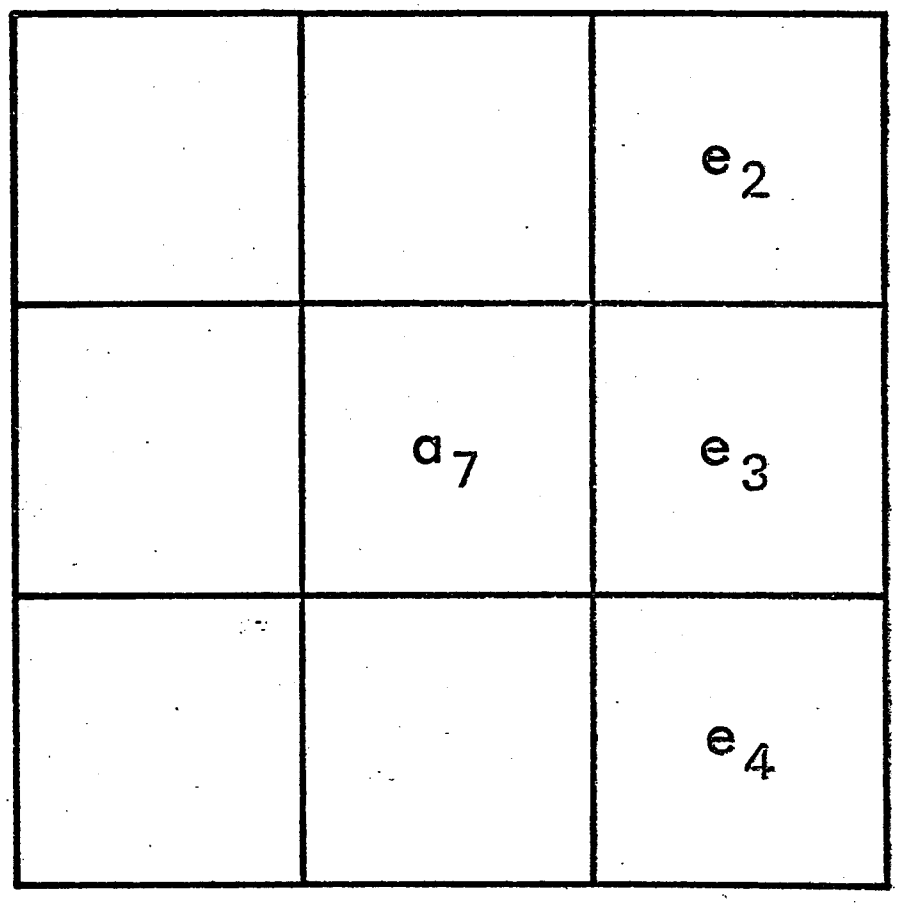

Figure 4 - Cells of possible movement from cell a_7.

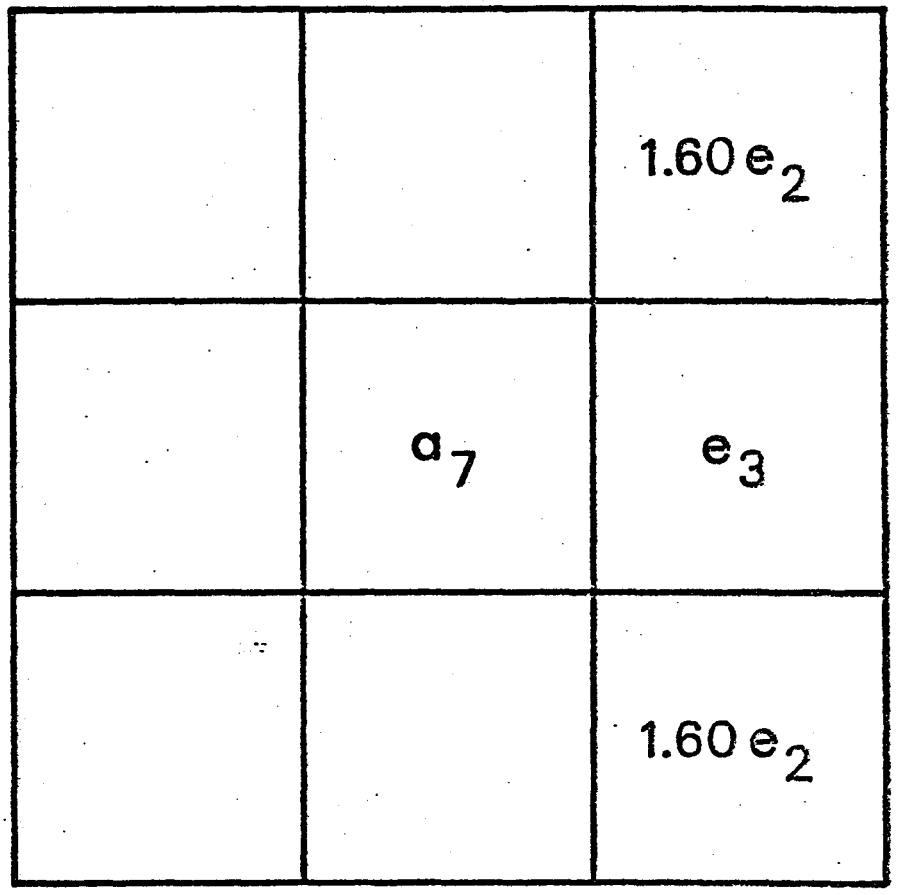

Figure 5 - Geometric corrections applied to cells.

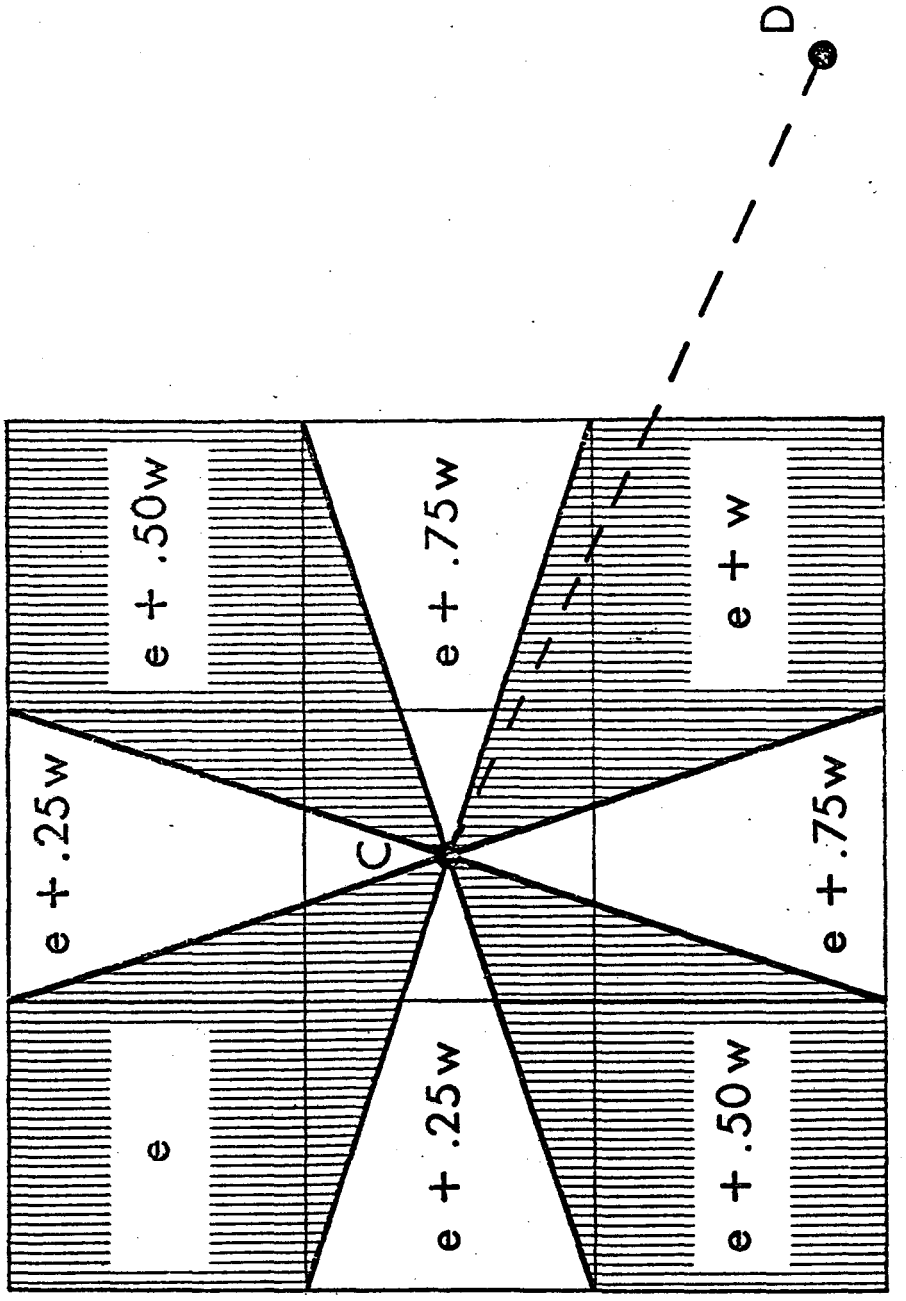

Figure 6 - Geometric relationship between a temporary end of path cell C and the end cell D with zones of directional bias weights delineated.

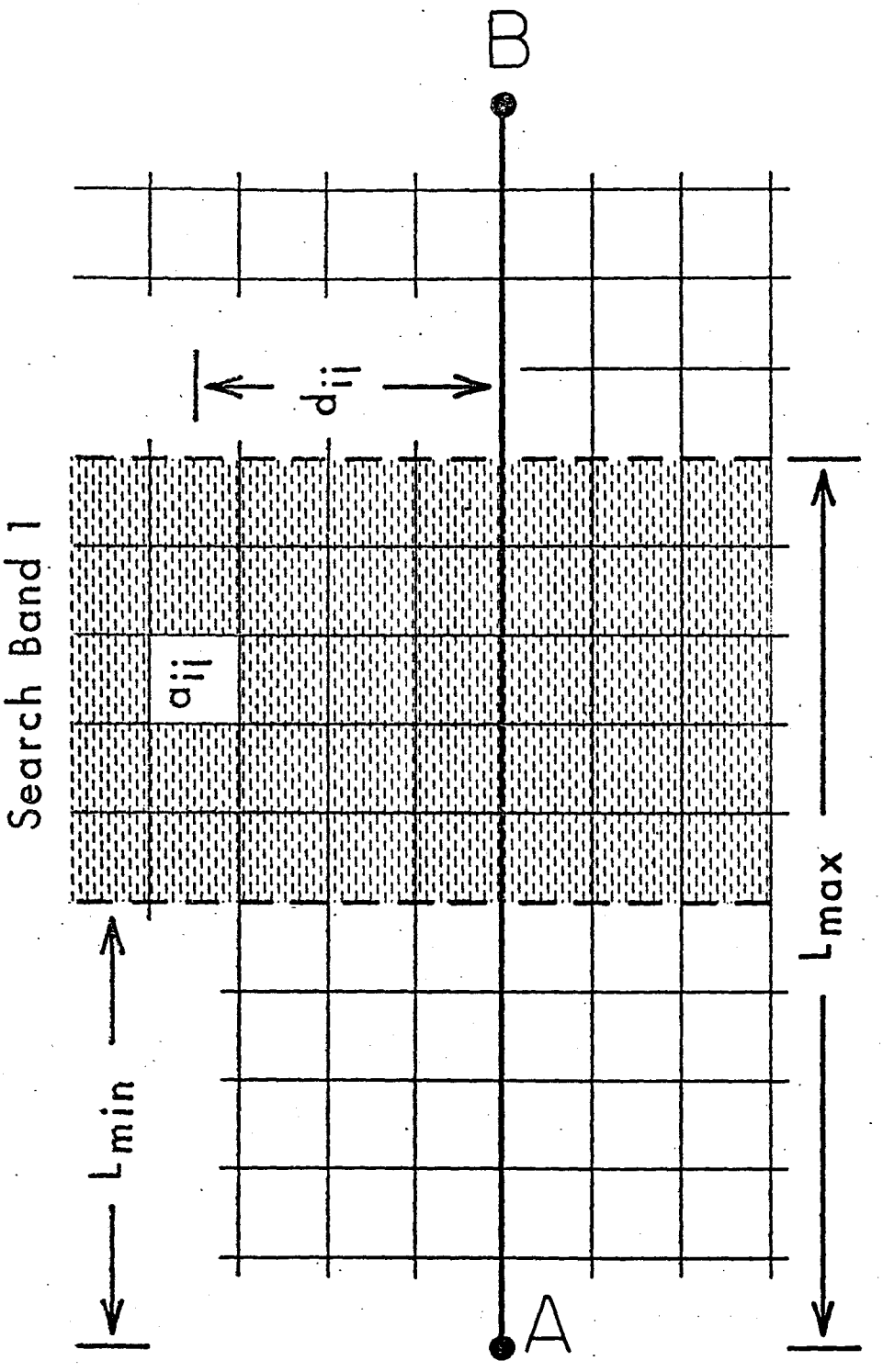

Figure 7 - Search band for selection of intermittant points between A and B.

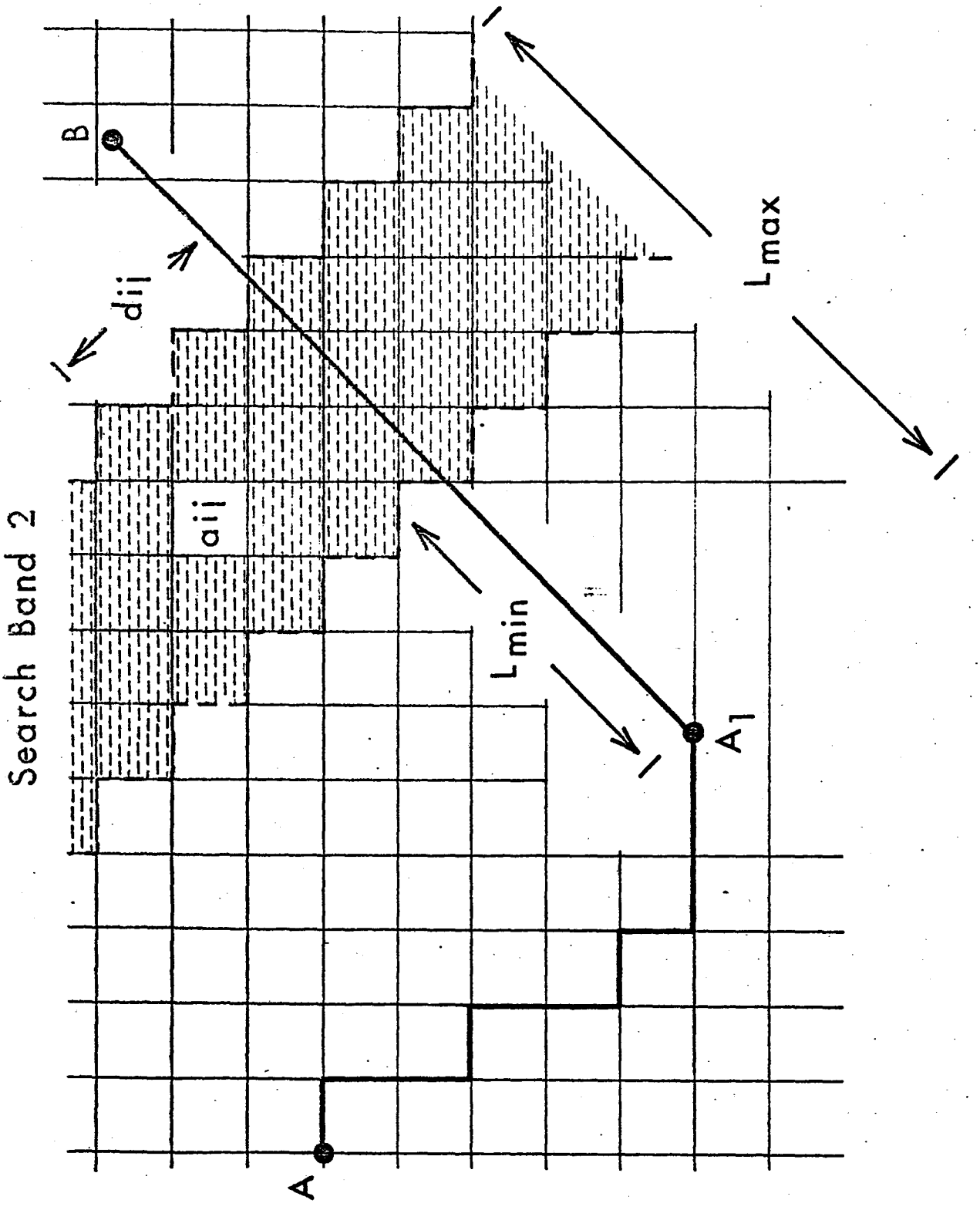

Figure 8 - Search band for a path nearly completed between A and B.

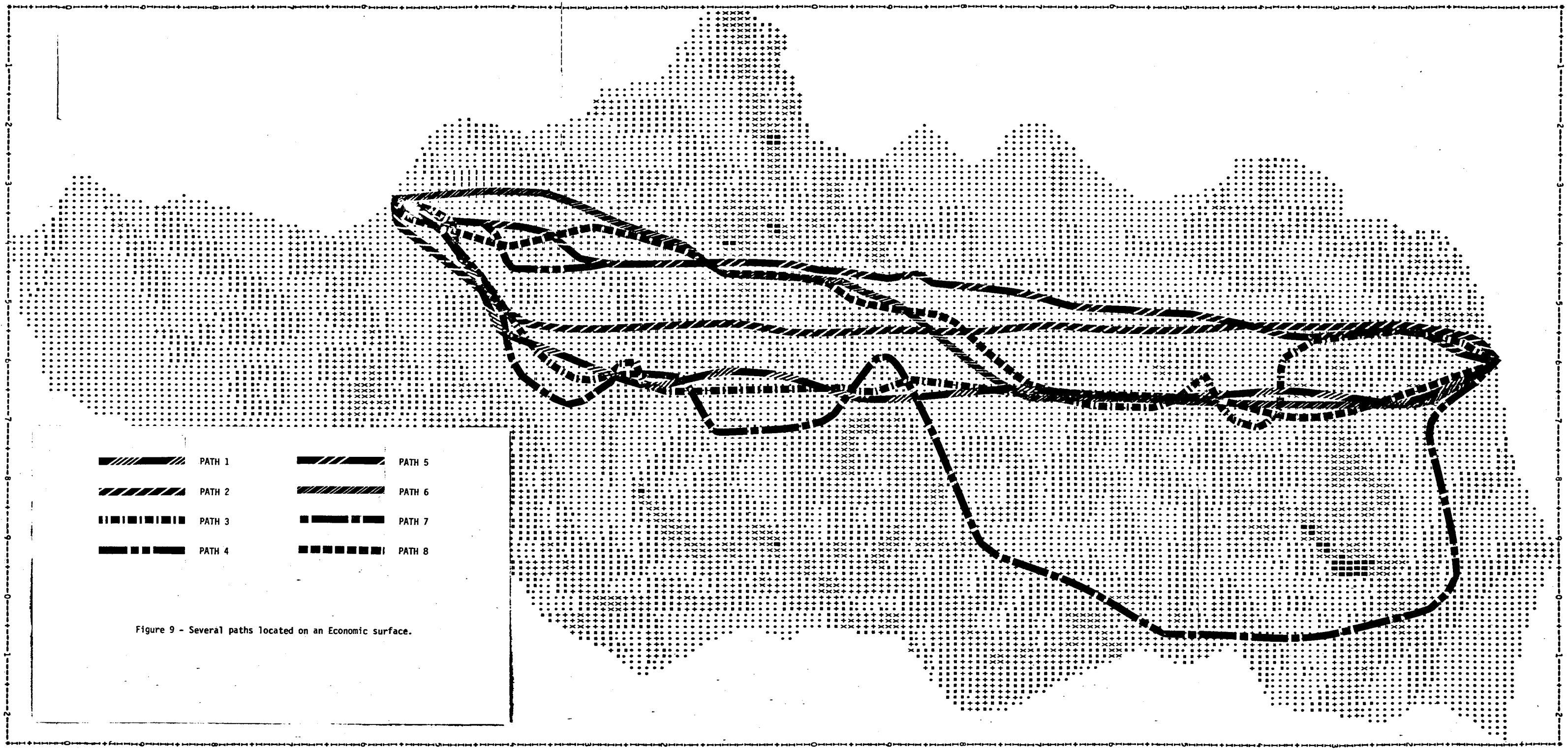

Figure 9 - Several paths located on an Economic surface.

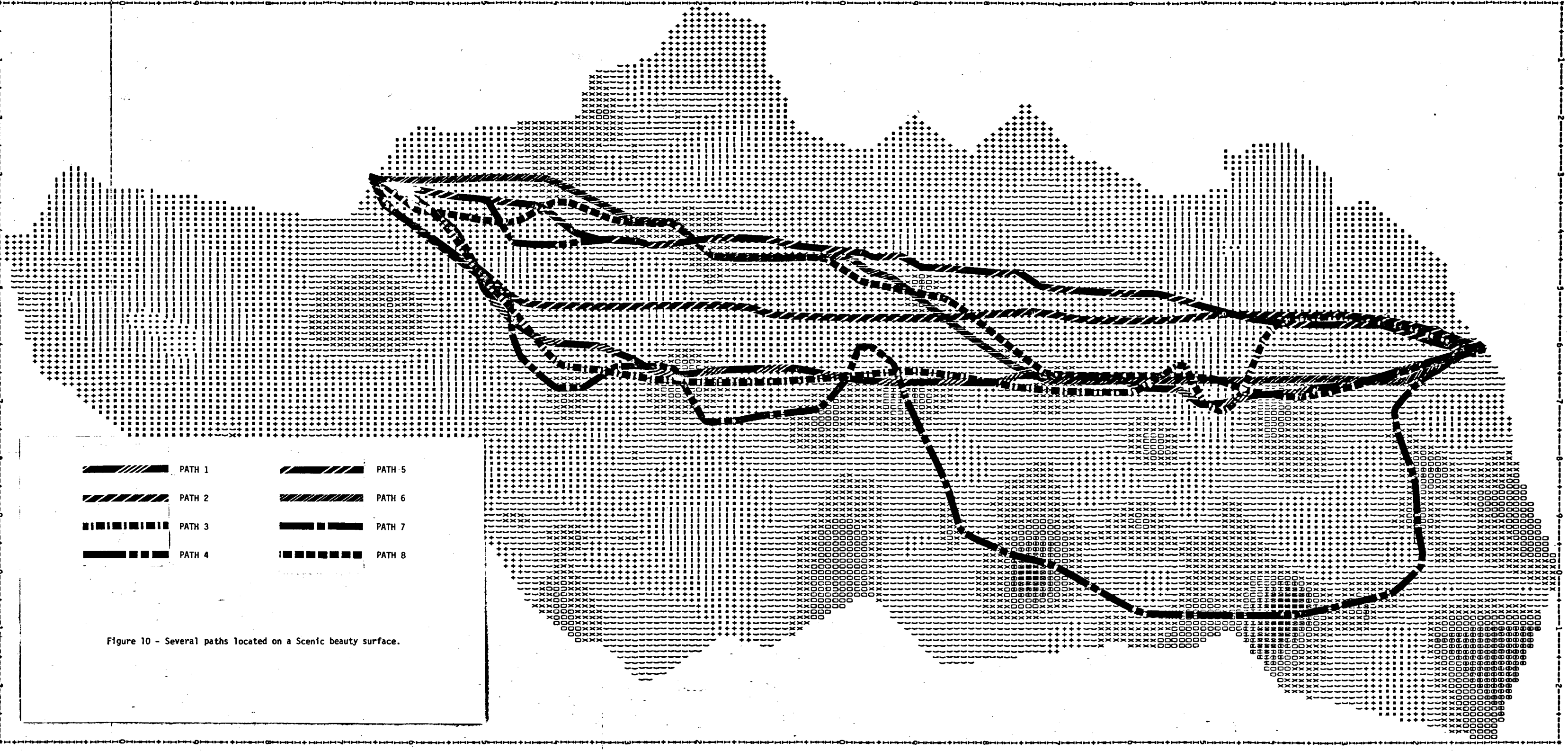

Figure 10 - Several paths located on a Scenic beauty surface.

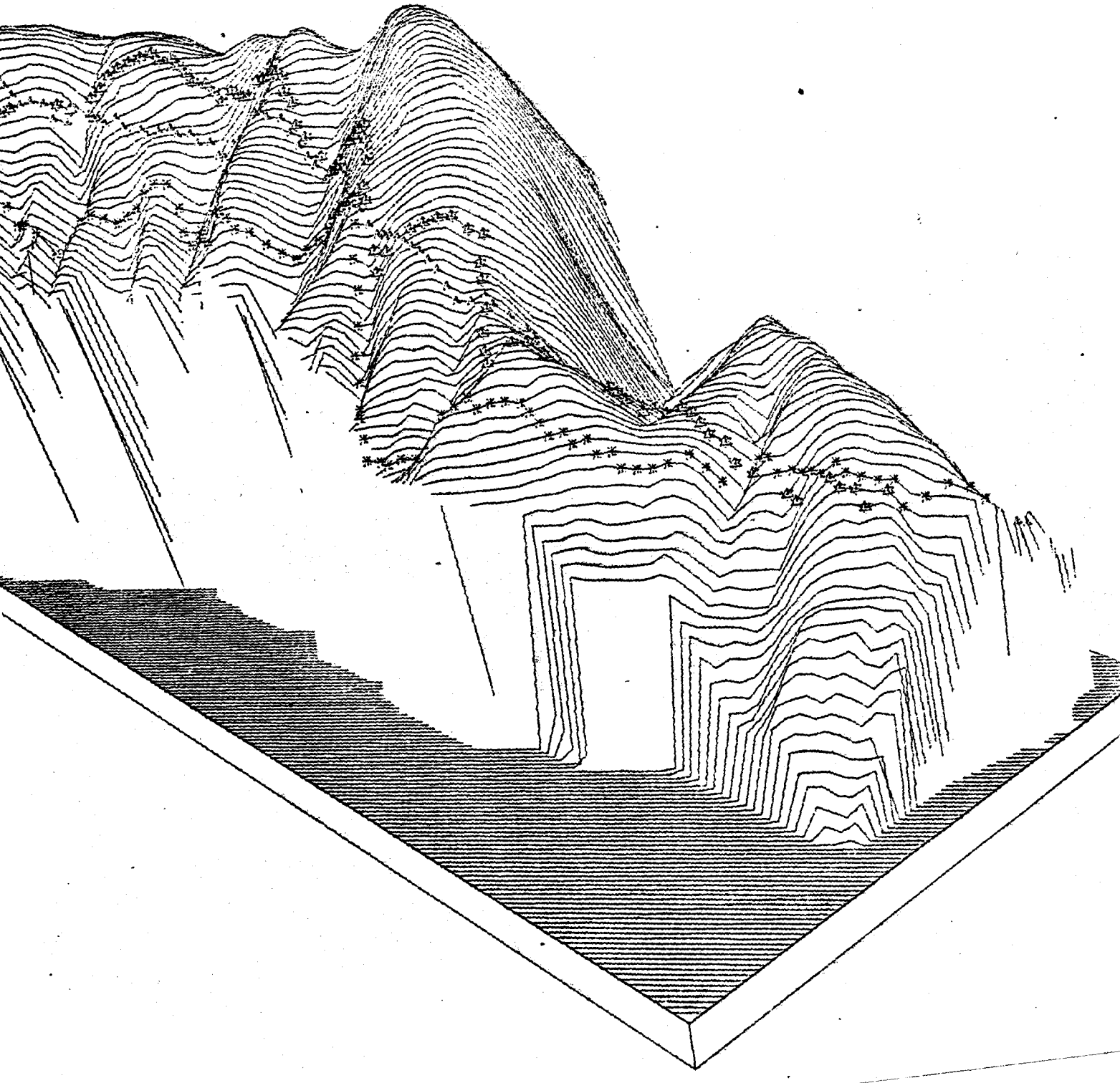

Figure 11 - Delineation of several paths on a perspective representation of the area.

PATH		SCENIC BEAUTY SURFACE		ECONOMIC SURFACE	
Number	How Defined	Sum of path cell values	Average path cell value	Sum of path cell values	Average path cell value
1	On economic surface with reduced absolute values using PATH.	209354.51	1006.5121	2227652.0	10,709.8654
2	On economic surface with PATH.	172214.74	967.4985	2004241.0	11,259.6293
3	On economic surface using TRAIL and low directional biasing weights.	251363.037	1001.4464	2836506.0	11,300.8207
4	On economic surface using TRAIL and medium directional biasing weights.	182176.529	990.0898	2404640.0	13,068.6951
5	On economic surface using TRAIL and high directional biasing weights.	182330.414	990.9262	2431580.0	13,215.1086
6	On scenic beauty surface using PATH.	192252.154	1006.5558	2258065.0	11,822.3298
7	On scenic beauty surface using low directional biasing weights.	357183.6720	1014.7263	4399988.0	12,499.9659
8	On scenic beauty surface using high directional biasing weights.	201496.9280	1007.4846	2630956.0	13,154.78

Table 1 - Comparison of average and total cost for several paths, half located to minimize construction cost and half located to maximize scenic quality.

Appendix A - Computer Program SLMATH

SLMATH

SLMATH has the ability to read data in several forms depending upon what the user has available. About the only real restriction on reading data with SLMATH is that the data must be preceeded by a record containing the number of rows, number of columns, minimum data value and maximum data value.

The preferred mode of reading data with SLMATH is a file that looks like one created with a SYMAP option 21 run (Dougenik and Sheehan, 1975): binary; 1st record containing number of rows, number of columns, number of columns per inch, number of rows per inch, minimum and maximum data values, followed by the data, in the order row 1, columns 1-130; row 2, columns 1-130; row n, columns 1-130; row 1, columns 131-260; row 2, columns 131-260; and so on. This data form will hereafter be referred to as SYMAP type.

SLMATH will also read coded data, depending on the value of 2 switches set by the user. This data may be in SYMAP form or, optionally, may not be. For data input of this type, the user must supply 2 fortran formats; one for the first record, containing min-max values, and one for the data itself.

SLMATH's purpose in life is to give printer maps of data in a matrix form. One form of operation of the program is printing a map from just one input file; i.e., a map of overstory density where a rectangular grid is superimposed over a resource map and each cell of the resulting matrix assigned a value depending on the overstory density within that cell. Another operational mode is to take two input maps and combine them in various ways to create a new map.

Input Data

The map input to SLMATH is expected to be on two files: TAPE 8, the incoming map (more on this later) and TAPE 11, the stored map. Unless the options described in the section coded data files are selected, the data is expected to be binary files in the form mentioned earlier. Also used by the program are INPUT, OUTPUT, TAPE 5, TAPE 6, TAPE 9, TAPE 10, and TAPE 12. If a map of just one data file is desired, the input is expected to be on TAPE 8.

Input off cards for scales and options is required, and takes only one card, plus comments, for most maps. The maps are divided into levels: these are n levels (n chosen by the user) between the maximum and minimum values. The symbols for these data levels are built in, but can be changed by selecting IPT = 1 (see table). In addition to regular simple symbol maps, overprint maps can be made, but there is no provision for changing the symbols used for this map type.

Card Input

CARD 1

Variable Name	Location (columns)	Type	Values (default)	Meaning
IMAP	1-2	Integer	1-2 (1)	number of input maps
LEVEL	3-4	"	1-50 (10)	number of levels desired on map (max 10 if overprint)
LMAP	6	"	1-4 (1)	see note 1
SCALE	7-16	Real	(1.0)	scale of data, in terms of cell width (note 2)
IPT	17	Integer	0-1 (0)	see card input order

CARD 1 continued

Variable Name	Location (columns)	Type	Values (default)	Meaning
IMP1	18-19	Integer	(0)	see the section coded data files for an explanation of these variables
IMP2	20-21	"	(0)	"
WT1	22-31	Real	(15)	see note 3
WT2	32-41	"	(.5)	"
ITRG	42	Integer	1-4 (1)	see note 4
FLIP	43	"	0-1 (0)	if 1, invert the incoming map (highest value becomes lowest) on combination map run
ILEV	44	"	0-50 (0)	see note 5

NOTE 1: There are 4 output options: LMAP = 1, symbol map with area summary (see output section); = 2, symbol map without summary; = 3, overprint map with summary; = 4, overprint without summary.

NOTE 2: A map with cells 100 meters wide would have a scale of (100) (39.37) / .1 = 39370.

NOTE 3: On a combination run for two maps, both maps can be weighted by a factor specified by the user. Both weights must be specified if either is to be, but if neither is they default to .5 (WT1 = TAPE 11, WT2, = TAPE 8).

NOTE 4: 4 functions for map combination currently exist; ITRG = 1, add maps; = 2, subtract maps; = 3, multiply maps; = 4, divide maps. In each case the stored map (TAPE 11) is operated on by the incoming map (TAPE 8) (eg., PT = (TAPE 11 * WT1) / (TAPE 8 * WT2)).

NOTE 5: In some cases, non linearly spaced levels may be desired setting ILEV to a number other than 0 will read ILEV upper bounds for checking data against. The FORMAT for the card containing the bounds is 8F10.2 (as many cards as necessary to get the levels will be read).

Card Input Order

With all the options available to the user, card input order for SLMATH can get complicated. Any card out of order will cause problems, as will a card left out. In the simple cases, card order is quite simple, but there are several card orders possible. The card order (with options specified) is:

Card 1, containing the variables in the table above	REQUIRED
Cards 2-5, containing formats for coded data (IMP1-IMP2)	OPTIONAL
Card 6, containing LEVEL symbols if IPT + 1 (SOA1)	OPTIONAL
Card 7, containing ILEV level upper bounds, if ILEV is greater than 0	OPTIONAL

```
      PROGRAM SLMATH(TAPE8,TAPE11,INPUT,OUTPUT,TAPE9,TAPE10,TAPE12,
     1                TAPE5=INPUT,TAPE6=OUTPUT)
      INTEGER BLANK,FLIP
      COMMON INP8,INP11,IMP1,IMP2,IPRTCR(10),PTS(130),STS(130),
     1       IPRT(4,131)
      DIMENSION ICHAR(10,4),IP(10,4),ISYM(50),IAREA(50),IL(55),JJ(10)
      DIMENSION RLEV(51),AREA(4,50)
      DIMENSION IPTS(130)
      EQUIVALENCE (PTS(1),IPTS(1))
      DATA BLANK,JROW,JCOL/' ',2*80000/
      DATA IP/40*' '/
      DATA IL/1,1,10,1,7,10,1,5,7,10,1,5,7,8,10,1,5,6,8,9,10,1,2,5,6,8,9
     1,10,1,2,5,6,7,8,9,10,1,2,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10/
      DATA IP(1,1),IP(2,1),IP(3,1),IP(4,1),IP(5,1),IP(6,1),IP(7,1)/1H.,
     11H-,1H=,1H+,1H*,1HX,1HO/
      DATA IP(8,1),IP(8,2),IP(9,1),IP(9,2),IP(10,1),IP(10,2),IP(10,3),
     1IP(10,4)/'O','-','O','X','O','X','A','V'/
      DATA ISYM/'0','1','2','3','4','5','6','7','8','9','A',
     1         'B','C','D','E','F','G','H','I','J','K','L','M','N','O',
     1          'P','Q','R','S','T','U','V','W','X','Y','Z','*','/','+',
     3          '-','=',':','(',')','\',',','.','<','>','$'/
      DATA IAREA/50*0/,AREA/200*0./,IBRD/'I'/
      ITSHR = 1
      IF(ITSHR .EQ. 0) CALL TSHR
      INP8 = 8
      INP11 = 11
      REWIND 9
      REWIND 10
      REWIND 12
C
C ***** SET ITSHR TO 0 IF RUNNING BATCH; 1 IF TIME SHARING
C
      READ(5,10) IMAP,LEVEL,LMAP,SCALE,IPT,IMP1,IMP2,WT1,WT2,ITRG,
     1           FLIP,ILEV
10    FORMAT(2I2,I1,F10.5,I1,2I2,2F10.5,2I1,I2)
      IF(IMP1 + IMP2 .GT. 0) CALL READR
C
C     SET SYMBOLS FOR THE BORDERS
C
      DO 5 I=1,10
5     IPRTCR(I) = ISYM(I)
C
C DIFFERENT SYMBOLS
```

Figure 1 - Computer Program Listing of SLMATH

```
C
      IF(IPT .GT. 0) READ(5,20) (ISYM(I),I=1,LEVEL)
20    FORMAT(50A1)
C
C BLANK THE PRINT ARRAY
C
      DO 30 I=1,131
       IPRT(1,I) = BLANK
       IPRT(2,I) = BLANK
       IPRT(3,I) = BLANK
30     IPRT(4,I) = BLANK
C
C SET DEFAULT VALUES FOR INPUT
C
      IF(ITRG .LT. 1) ITRG = 1
      IF(WT1  .GT. 0) GO TO 40
      WT1 =  .5
      WT2 = .5



C
40    READ(INP8) IROW,ICOL,DR,DC,VMIN,VMAX
      INPCO = INP8
      IF(IMAP .EQ. 1) GO TO 50
C
C INVERT MAP ?
C
      IF(FLIP .GT. 0) GO TO 55
      DUM = VMIN
      VMIN = -VMAX
      VMAX = -DUM
C
55    READ(INP11) JROW,JCOL,CR,CC,VMIN,VMAX
      IF(JROW-IROW + JCOL-ICOL .EQ. 0) GO TO 50
      WRITE(6,1010) JROW,JCOL,IROW,ICOL
1010  FORMAT(1H ,'DATA NOT COMPATBLE'/' PREVIOUS STORED MAP HAS'
     1    ,I5,' ROWS AND ',I5,' COLUMNS.'/' CURRENT MAP HAS',I5,
     2  '  ROWS AND ',I5,' COLUMNS')

50    NROW = MINO(IROW,JROW)
      NCOL = MINO(ICOL,JCOL)
      NTIME = 1
```

Figure 1 - continued

```
        JJ(1) = NCOL
        IF(NCOL .LE. 130) GO TO 70
        NTIME = NCOL / 130
        DO 60 I=1,NTIME
60      JJ(I) = 130
        JJ(NTIME + 1) = NCOL - (NTIME *130)
        IF(NTIME * 130 .LT. NCOL) NTIME = NTIME +1
70      IF(IMAP .EQ. 1) GO TO 210
90      IF(FLIP .GT. 0) WT1 = -WT1
C
C PERFORM DESIRED FUNCTION MANIPULATION ON COMBINED MAPS
C
        DO 200 I=1,NTIME
         J = JJ(I)
         DO 200 K=1,NROW
        IF (I.NE.1) GO TO 96
          IF(K .LE. IROW) READ(INP8)(STS(L),L=1,J)
          IF(K .LE. JROW) READ(INP11)(PTS(L),L=1,J)
        GO TO 97
 96     CONTINUE
          IF(K .LE. IROW) READ(INP8) DDUM,(STS(L),L=1,J)
          IF(K .LE. JROW) READ(INP11) DDUM,(PTS(L),L=1,J)
 97     CONTINUE
100       GO TO (110,130,150,170),ITRG
C
110       DO 120 L=1,J
           IF(IPTS(L) .EQ. 25) GO TO 120
           PTS(L) = PTS(L) * WT1 + STS(L) * WT2
           IF(PTS(L) .LT. VMIN) VMIN = PTS(L)
           IF(PTS(L) .GT. VMAX) VMAX = PTS(L)
120       CONTINUE
          GO TO 190
C
130       DO 140 L=1,J
           IF(IPTS(L) .EQ. 25) GO TO 140
           PTS(L) = PTS(L) * WT1 - STS(L) * WT2
           IF(PTS(L) .LT. VMIN) VMIN = PTS(L)
           IF(PTS(L) .GT. VMAX) VMAX = PTS(L)
140       CONTINUE




          GO TO 190
```

Figure 1 - continued

```
C
150         DO 160 L=1,J
             IF(IPTS(L) .EQ. 25) GO TO 160
             PTS(L) = PTS(L) * WT1 * STS(L) * WT2
             IF(PTS(L) .LT. VMIN) VMIN = PTS(L)

             IF(PTS(L) .GT. VMAX) VMAX = PTS(L)
160         CONTINUE
            GO TO 190
C
170         DO 180 L=1,J
             IF(IPTS(L) .EQ. 25) GO TO 180
             PTS(L) = (PTS(L) * WT1) / (STS(L) *WT2)
             IF(PTS(L) .LT. VMIN) VMIN = PTS(L)
             IF(PTS(L) .GT. VMAX) VMAX = PTS(L)
180         CONTINUE
C
190         WRITE(10) (PTS(L),L=1,J)
200       CONTINUE
          REWIND 10
          INPCO = 10
C
C COMPUTE LEVELS AND THEIR SPACING
C
210       VSUM = (VMAX - VMIN) / LEVEL
          IF(ILEV .LE. 0) GO TO 230
C
C IF NON-LINEARLY SPACED LEVELS ARE DESIRED, READ THEM NOW
C
          READ(5,220)(RLEV(I),I=1,ILEV)
220       FORMAT(8F10.5)
          LEVEL = ILEV
          GO TO 250
C
C SET UP POINTERS TO OVERLAY MAP SYMBOLS
C
230       DUM = VMIN
          DO 240 I=1,LEVEL
          DUM = DUM + VSUM
240       RLEV(I) = DUM
250       IF(LMAP .LE. 2) GO TO 902
          J = 0
          K = LEVEL - 1
          DO 260 I=1,K
260       J = J + I
```

Figure 1 - continued

```
C
        DO 270 I=1,LEVEL
         L = IL(J + I)
         ICHAR(I,1) = IP(L,1)
         ICHAR(I,2) = IP(L,2)
         ICHAR(I,3) = IP(L,3)
270     ICHAR(I,4) = IP(L,4)
C
C PRINT OVERLAID MAP
C
      DO 41 I=1,NTIME
      J=JJ(I)
         N = J + 1
        CALL BORDER(I,J,'1')
      DO 42 K=1,NROW




        IF (I.EQ.1) READ(INPCO) (PTS(L),L=1,J)
        IF (I.NE.1) READ(INPCO) DDUM,(PTS(L),L=1,J)
      DO 48 L=1,4
      DO 48 M=1,130
      IPRT(L,M)=BLANK
   48 CONTINUE
C
      DO 43 L=1,J
      IF(IPTS(L) .EQ. 25) GO TO 43
   44 DO 46 M=1,LEVEL
      IF(PTS(L) .LE. RLEV(M)) GO TO 47
   46 CONTINUE
      M=LEVEL
   47 IPRT(1,L)=ICHAR(M,1)
      IPRT(2,L)=ICHAR(M,2)
      IPRT(3,L)=ICHAR(M,3)
      IPRT(4,L)=ICHAR(M,4)
        IAREA(M) = IAREA(M) + 1
        ATOT = ATOT + 1.
   43 CONTINUE
C
C BORDER CHANGE FROM 'I' AT INCREMENTS DIVISIBLE BY 5 AND 10
C
         M = K / 5
         IF(M * 5 .NE. K) GO TO 101
```

Figure 1 - continued

```
          IBRD = '+'
          M = M / 2
          IF(M * 10 .NE. K) GO TO 101
          M = MOD(M,10) + 1
          IBRD = IPRTCR(M)
101      IPRT(1,N) = IBRD
C
       WRITE(6,105) IBRD,(IPRT(1,M),M=1,N)
       WRITE(6,106) (IPRT(2,M),M=1,J)
       WRITE(6,106) (IPRT(3,M),M=1,J)
       WRITE(6,106) (IPRT(4,M),M=1,J)
105      FORMAT(1H ,132A1)
106      FORMAT(1H+,1X,131A1)
         IBRD = 'I'
    42 CONTINUE
         CALL BORDER(I,J,BLANK)
    41 CONTINUE
       GO TO 49
C
C PRINT NON-OVERLAID MAP
C
 902   DO 941 I=1,NTIME
       J=JJ(I)
         N = J + 1
         CALL BORDER(I,J,'1')
       DO 942 K=1,NROW
         IF (I.EQ.1) READ(INPCO) (PTS(L),L=1,J)
         IF (I.NE.1) READ(INPCO) ODUM,(PTS(L),L=1,J)
       DO 943 L=1,J
       IPRT(1,L)=BLANK
       IF(IPTS(L) .EQ. 25) GO TO 943
 944   DO 946 M=1,LEVEL
       IF(PTS(L) .LE. RLEV(M)) GO TO 947
 946   CONTINUE
       M=LEVEL
 947   IPRT(1,L)=ISYM(M)



       IAREA(M)=IAREA(M)+1
         ATOT = ATOT + 1
 943   CONTINUE
          M = K / 5
```

Figure 1 - continued

```
          IF(M * 5 .NE. K) GO TO 940
          IBRD = '+'
          M = M / 2
          IF(M * 10 .NE. K) GO TO 940
          M = MOD(M,10) + 1
          IBRD = IPRTCR(M)
940       IPRT(1,N) = IBRD
        WRITE(6,105) IBRD,(IPRT(1,M),M=1,N)
          IBRD = 'I'
 942    CONTINUE
          CALL BORDER(I,J,BLANK)
941       CONTINUE
C
C READ HOWEVER MANY COMMENTS THERE ARE
C
    49 WRITE(6,91)
91        FORMAT(////)
92        READ(5,93)(STS(I),I=1,16)
93        FORMAT(16A5)
C
C    CHECK FOR END OF COMMENTS
C
          IF(STS(1) .EQ. 5HEND  ) GO TO 95
          WRITE(6,94)(STS(I),I=1,16)
94        FORMAT(1X,16A5)
          GO TO 92
C
C SET UP FOR BEGINNING OF SUMMARY
C
95        WRITE(6,300) VMIN,VMAX
300       FORMAT(////,' ABOVE MAP HAS A MINIMUM OF:',F10.3,/
     1           ,17X,'MAXIMUM OF:',F10.3)
          IF(FLIP .GT. 0) WRITE(6,310)
310       FORMAT(' INCOMING MAP WAS INVERTED')
          IF(MOD(LMAP,2) .EQ. 0) GO TO 800
C
C ASQ IS SCALER FOR RECTANGULAR CELLS (.10 WIDE BY .125 TALL)
C DO 917 COMPUTES AREAS IN ACRES, SQ. MILES, HECTARES, AND PERCENT
C  (RESPECTIVELY) FOR AREA SUMMARY IF DESIRED
C
        ASQ=(SCALE*SCALE)/(144.*80.)
        DO 917 I=1,LEVEL
        AREA(1,I)=(IAREA(I)*ASQ)/43560.
        AREA(3,I)=AREA(1,I)/640.
        AREA(2,I)=(IAREA(I)*ASQ)/107600.
```

Figure 1 - continued

```
917   AREA(4,I)=(IAREA(I)/ATOT)*100.
C
C SET UP FOR MAX AND MIN PRINTING
C
800      L = LEVEL + 2
         DO 916 I=1,LEVEL
          K = L - I
          J = K - 1
916       RLEV(K) = RLEV(J)
         RLEV(1) = VMIN
         ML = -9
         IH = 0



C
C SET UP DATA LEVEL OUTPUT FOR BOTH TYPES OF MAP
C
810      ML = ML + 10
         IH = IH + 10
C
C IF IH AND ML ARE = OR > LEVEL, WE'RE DONE
C
         IF(IH .LE. LEVEL) GO TO 820
          IF(ML .GT. LEVEL) GO TO 999
          IH = LEVEL
820      IF(LMAP .LT. 3) GO TO 830
C
C OVER PRINT SETUP FOR PRINTING SYMBOLS FOR LEVELS (9 SYMBOLS,
C  1 BLANK)
C
       K1=0
       DO 520 I=1,LEVEL
       DO 521 J=1,9
       K1=K1+1
       IPRT(1,K1)=ICHAR(I,1)
       IPRT(2,K1)=ICHAR(I,2)
       IPRT(3,K1)=ICHAR(I,3)
       IPRT(4,K1)=ICHAR(I,4)
  521  CONTINUE
       K1=K1+1
       IPRT(1,K1)=BLANK
       IPRT(2,K1)=BLANK
```

Figure 1 - continued

```
      IPRT(3,K1)=BLANK
520   IPRT(4,K1) = BLANK
        GO TO 840
C
C NON OVER-PRINT SYMBOL SETUP
C
830     K1 = 0
        DO 923 I=ML,IH
         DO 924 J=1,9
          K1 = K1 + 1
924       IPRT(1,K1) = ISYM(I)
         K1 = K1 + 1
923      IPRT(1,K1) = BLANK
C
C PRINT MAX AND MIN
C
840     L = ML + 1
        M = IH + 1
        WRITE(6,912)
912     FORMAT(1HO,'  VALUE RANGE FOR EACH LEVEL')
        WRITE(6,913)(RLEV(I),I=ML,IH)
913     FORMAT(1HO,' MINIMUM',10F10.3)
        WRITE(6,914) (RLEV(I),I=L,M)
914     FORMAT(1HO,' MAXIMUM',10F10.3)
C
C SKIP AREA SUMMARY IF IT'S NOT DESIRED
C
        IF(MOD(LMAP,2) .EQ. 0) GO TO 850
      WRITE(6,915) (IAREA(I),I=ML,IH)
915     FORMAT(1HO, ' CELLS  '   ,10I10)
      WRITE(6,918) (AREA(1,I),I=ML,IH)
918     FORMAT(1HO, ' ACREAGE',10F10.2)
      WRITE(6,919) (AREA(2,I),I=ML,IH)




919     FORMAT(1HO, ' HECTARS',10F10.2)
      WRITE(6,920)(AREA(3,I),I=ML,IH)
920     FORMAT(1HO, ' SQ.MILES',F9.4,9F10.4)
      WRITE(6,921) (AREA(4,I),I=ML,IH)
921     FORMAT(1HO, ' PERCENT',10F10.2)
C
C PRINT SYMBOLS FOR LEVELS
```

Figure 1 - continued

```
C
850      WRITE(6,860)
860      FORMAT(1H0,'  SYMBOLS FOR DATA LEVELS')
         DO 532 I=1,5
          WRITE(6,533) (IPRT(1,J),J=1,K1)
          WRITE(6,534) (IPRT(2,J),J=1,K1)
          WRITE(6,534) (IPRT(3,J),J=1,K1)
532       WRITE(6,534) (IPRT(4,J),J=1,K1)
533      FORMAT(1H ,9X,100A1)
534      FORMAT(1H+,9X,100A1)
C
C NOW GO AND SEE IF THERE ARE MORE THAN 10 LEVELS
C
         GO TO 810
C
C  FINISH IT OFF
C
999      DC = ATOT * ASQ
         CC = DC / 43560.
         CR = DC / 107600.
         DC = CC / 640.
         WRITE(6,1000) ATOT,CC,CR,DC
1000     FORMAT(//,' TOTAL AREAS:',F7.0,' CELLS',F10.2,' ACRES',F10.2,
     1          ' HECTARES',F10.2,' SQUARE MILES')
          WRITE(6,9050)
9050     FORMAT(///,1X,'THIS MAP PREPARED BY THE LABORATORY FOR REMOTE ',
     !        'SENSING AND COMPUTER MAPPING.')
         REWIND INPCO
         REWIND 12
         IF(IABS(IMP1 + IMP2) .EQ. 0 .AND. IMAP .EQ. 1)
     1                    GO TO 710
         STS(1) = .08
         STS(2) = .10
         WRITE(12) NROW,NCOL,STS(1),STS(2),VMIN,VMAX
         DO 700 I=1,NTIME
          J = JJ(I)
          DO 700 K=1,NROW
         IF (I.EQ.1) READ(INPCO) (PTS(L),L=1,J)
         IF (I.NE.1) READ(INPCO) DDUM,(PTS(L),L=1,J)
           WRITE(12) (PTS(L),L=1,J)
700        CONTINUE
C
         WRITE(6,720)
720      FORMAT(' END COMPLEX RUN - NEW DATA FILE WRITTEN TO TAPE12')
C
```

Figure 1 - continued

```
710     IF(ITSHR .EQ. 0) CALL CLSE
        STOP
        END
        SUBROUTINE READR
        COMMON INP(2),IMP1,IMP2,IFORM(16),PTS(774)
        EQUIVALENCE (INP1,INP(1)),(INP(2),INP2)
        DIMENSION NCOL(10)
        DATA NCOL/10*0/,ZERO/0.0/
C
C GO TO 80 IF DATA NOT IN SYMAP FORM
C
        IF(IABS(IMP1) .GT. 1) GO TO 80
        ISWT = 1
        IADD = 1
        IF(IMP2 .EQ. 0) IADD = 2
C
C READ FORMAT FOR THE INITIAL MIN - MAX RECORD
C
15      READ(5,10) IFORM
10      FORMAT(16A5)
        INPT = INP(ISWT)
        READ(INPT,IFORM) IROW,ICOL,VMIN,VMAX
        INPCO1 = INP(ISWT) + 1
        WRITE(INPCO1) IROW,ICOL,ZERO,ZERO,VMIN,VMAX
C
C FORMAT FOR READING THE BODY OF THE DATA
C
        READ(5,10) IFORM
        NT = ICOL / 130
        IF(NT * 130 .LT. ICOL) NT = NT + 1
        NCOL(NT) = ICOL - (NT - 1) * 130
C
C HERE WE'LL READ THE CODED DATA BASE (ASSUMED TO BE IN SYMAP FORM)
C NOTE THAT THE FORMAT INPUT IS SUPPOSED TO DO ANY JUMPING ABOUT
C  FOR PICKING UP THE CORRECT DATA POINTS IN THE DATA BASE
C
        DO 20 I=1, NT
         J = NCOL(I)
         DO 20 K=1,IROW
          READ(INPT,IFORM)(PTS(L),L=1,J)
```

Figure 1 - continued

```
20          WRITE(INPCO1) (PTS(L),L=1,J)
C
25        INP(ISWT) = INPCO1
          REWIND INPCO1
          ISWT = ISWT + IADD
          IF(ISWT .GT. 2) RETURN
          GO TO 15
C
C COME HERE FOR NON SYMAP TYPE DATA. IT'S NECESSARY TO HANDLE THIS
C  DATA DIFFERENTLY BECAUSE OF THE LIMITATIONS IN THE MAINLINE PROGRAM
C NOTE THAT THE CASE OF ICOL <= 130 IS A TRIVIAL CASE OF THE NON SYMAP
C  CASE, HANDLED BY THE SYMAP FORM CODE ABOVE
C
30        K = 0
          DO 40 I=1,NT
           K = K + 130
40         NCOL(I) = K
C
          DO 60 I=1,IROW
           READ(INPT,IFORM)(PTS(J),J=1,ICOL)
           WRITE(INPCO1)(PTS(J),J=1,130)
           DO 50 J=2,NT
            M = NCOL(J-1) + 1
            N = NCOL(I)
50          WRITE(10) J,(PTS(K),K=M,N)
60         CONTINUE
C
C NOW READ DATA FROM THE TAPE WHERE IT WAS STORED (TAPE10) AND GET IT
C  TO THE DATA TAPE IN THE CORRECT ORDER
C
          NCOL(NT) = ICOL
          DO 70 I=2,NT
           J = NCOL(I) - (I-1) * 130
           REWIND 10
           DO 75 K=1,IROW
            READ(10) L,(PTS(M),M=1,130)
            IF(L .NE. I) GO TO 80
            WRITE(INPCO1) (PTS(M),M=1,J)
75          CONTINUE
70         CONTINUE
```

Figure 1 - continued

```
        REWIND 10
        GO TO 25
C
C COME HERE FOR BOTH MAPS FROM THE SAME DATA BASE
C
80      INP1 = 9
        INP2 = 12
        READ(5,10) IFORM
        IF(IMP2 .LE. 0) GO TO 90
        INP1 = 12
        INP2 = 9
90      READ(8,IFORM) IROW,ICOL,(PTS(I),I=1,4)
        WRITE(INP1) IROW,ICOL,ZERO,ZERO,PTS(1),PTS(2)
        WRITE(INP2) IROW,ICOL,ZERO,ZERO,PTS(3),PTS(4)
        READ(5,10) IFORM
        NT = ICOL / 130
        IF(NT * 130 .LT. ICOL) NT = NT + 1
        IF(IMP1 .LT. 0) GO TO 120
        NCOL(NT) = ICOL - (NT -1) * 130
         DO 100 I=1,NT
         J = NCOL(I) * 2
         DO 110 K=1,IROW
          READ(8,IFORM)(PTS(L),L=1,J)
          WRITE(INP1)(PTS(L),L=1,J,2)
          WRITE(INP2)(PTS(L),L=2,J,2)
110      CONTINUE
100     CONTINUE
C
        REWIND INP1
        REWIND INP2
        RETURN
C
C HERE IF FILE IN NON SYMAP FORM AND BOTH MAPS FROM SAME DATA BASE
C
120     K = 0
        DO 130 I=1,NT
         NCOL(I) = K
130     K = K + 130
C
        K = ICOL * 2
        DO 140 I=1,IROW

         READ(8,IFORM)(PTS(J),J=1,K)
         WRITE(INP1)(PTS(J),J=1,260,2)
         WRITE(INP2)(PTS(J),J=2,260,2)
```

Figure 1 - continued

```
          DO 150 J=2,NT
           M = NCOL(I-1) * 2 + 1
           N = NCOL(I) * 2
           WRITE(10) J,(PTS(L),L=M,N,2)
           M = M + 1
150       WRITE(11) J,(PTS(L),L=M,N,2)



140       CONTINUE
C
          NCOL(NT) = IROW
          DO 160 I=2,NT
           REWIND 10
           REWIND 11
          K = NCOL(I) - NCOL(I - 1)
           DO 180 J=1,IROW
            READ(10) M,(PTS(L),L=1,130)
            IF(M .NE. I) GO TO 170
            WRITE(INP1) (PTS(L),L=1,K)
170         READ(11) M,(PTS(L),L=1,130)
            IF(M .NE. I) GO TO 180
            WRITE(INP2) (PTS(L),L=1,K)
180        CONTINUE
160       CONTINUE
C
          REWIND INP1
          REWIND INP2
          REWIND 10
          RETURN
          END
          SUBROUTINE BORDER(IST,IEND,IFST)
C
C THIS ROUTINE PUTS AN UPPER OR LOWER BORDER AROUND A MAP.
C THE BORDER WILL HAVE THE COUMNS NUMBERED ACCORDING TO:
C         IST      HAS THE NUMBER THE NUMBERING STARTS
C         IEND     WHERE THE NUMBERING ENDS
C         IFST     THE CHARACTER TO BE PRINTED IN THE FIRST PRINT LOCATION
C
          COMMON IDUM(4),ISYMB(10),IPRT(132)
          DATA IS,IDASH,IP/'*','-','+'/

          DO 10 I=1,132
```

Figure 1 - continued

```
10      IPRT(I) = IDASH

        DO 20 I=1,130,5
20      IPRT(I) = IP

        IST1 = MOD(IST *13 -13,10) + 1
        DO 30 I=1,IEND,10
         IPRT(I) = ISYMB(IST1)
30       IST1 = MOD(IST1,10) + 1

        IE = IEND + 2
        IPRT(1) = IS
        IPRT(IE) = IS
        WRITE(6,40) IFST,(IPRT(I),I=1,IE)
40      FORMAT(133A1)
        RETURN
        END
```

Figure 1 - continued

Appendix B - Computer Program TRACK

TRACK

This program is designed to compute a series of coordinates which represent a path manually traced with a digitizing cursor across a computer printer map located on a digitizing tablet. The scale of the map placed on the tablet and number of rows per inch are taken into account by questions asked the user, as such any map may be used.

The program is interactive. By looking at the questions asked in the code the user should be aware of what is required in it's operation. The program begins by assuming a printer map has been placed on the digitizing tablet. The number of rows and columns between four corner points (input to the program) allows cell coordinates to be computed for a path across the map in terms of the cells traversed by the path. Once the rows and columns are input the user is then asked to digitize the path across the map. This is done by inputting as many points as deemed necessary to define the path. The program assembles a series of cell coordinates representing the cells traversed.

The traversed cells are assigned values of 1.0 in the matrix M (rows, columns). All other cells are assigned values of 0.0. The user is then told the number of cells in the path and asked how many maps he wishes to superimpose this path on. The data values of the maps on which the path is to be placed are assumed to reside in files FOR61.DAT, FOR62.DAT, etc. If only one map, it is on FOR61.DAT. If two maps, the second would be on FOR62.DAT. The data on these files are assumed to be in SYMAP standard format (see discussion of SLMATH).

The program then picks out the cells in the path from the data arrays on FOR61.DAT and computes the sum of all values in the path as well as the average value for a cell in the path. These are displayed to the user, and then repeated for as many map data files as were linked to the program via the FOR6n.DAT files.

```
      DIMENSION X(200),Y(200),M(300,140),RVAL(300,140)
      IBUG=1
      TOL=99.99
C.....GET BORDER OF MAP; DETERMINE MIN AND MAX X AND Y.....
      WRITE(5,3001)
 3001 FORMAT(1X,'TAP IN 4 CORNERS OF MAP')
      XMIN=99.0
      XMAX=0.0
      YMIN=99.0
      YMAX=0.0
      DO 900 I=1,4
      READ(24,3002) XPOINT,YPOINT
      IF (IBUG.EQ.1) WRITE(32,3999) XPOINT,YPOINT
 3999 FORMAT(1X,2F10.2)
 3002 FORMAT(1X,2F4.2)
      IF (XPOINT.LT.XMIN) XMIN=XPOINT
      IF (XPOINT.GT.XMAX) XMAX=XPOINT
      IF (YPOINT.LT.YMIN) YMIN=YPOINT
      IF (YPOINT.GT.YMAX) YMAX=YPOINT
 900  CONTINUE
C.....GET # ROWS & COLUMNS.....
      WRITE(5,3003)
 3003 FORMAT(1X,'ENTER NUMBER OF ROWS AND COLUMNS ',
     1   '(SEPARATED BY A SPACE):')
      READ(24,3004) NROW,NCOL
 3004 FORMAT(2I)
      NPOINT=0
C.....READ IN POINTS ON PATH.....
 105  WRITE(5,3005)
 3005 FORMAT(1X,'TAP IN POINTS OF THE PATH, MOVING FROM ONE ',
     1   'END',/,5X,'TO THE OTHER (ENTER "99999" TO STOP)')
 110  READ(24,3006) IA
 3006 FORMAT(A1,2I4)
      IF (IA.EQ.1HT) GO TO 500
      IF (IA.EQ.1H*) GO TO 510
      REREAD 3007, ENDIT
 3007 FORMAT(1F)
      IF (ENDIT.EQ.99999.) GO TO 600
      WRITE(5,3008)
 3008 FORMAT(1X,'INVALID ENTRY:')
      GO TO 105
 500  REREAD 3002, TOL
      GO TO 110
 510  NPOINT=NPOINT +1
      REREAD 3002, X(NPOINT),Y(NPOINT)
```

Figure 1 - Computer Program Listing of TRACK

```
      IF (NPOINT.EQ.1) GO TO 110
      P1=X(NPOINT-1) - X(NPOINT)
      P2=Y(NPOINT-1) - Y(NPOINT)
      P1=ABS(P1)
      P2=ABS(P2)
      IF (P1.LE.TOL.AND.P2.LE.TOL) GO TO 110
      WRITE(5,4002) X(NPOINT),Y(NPOINT),X(NPOINT-1),Y(NPOINT-1)
 4002 FORMAT(1X,'POINT ',F5.2,',',F5.2,' OUT OF TOLERANCE-',
     1   'DISCARDED',/,10X,'(LAST POINT WAS ',
     2   F5.2,',',F5.2,')')
      NPOINT=NPOINT-1
      GO TO 110
 600  CONTINUE
C.........DETERMINE SIZE OF EACH CELL: CSPAN=WIDTH(COLUMN-WISE),
C............RSPAN=LENGTH(ROW-WISE)
      IF (IBUG.NE.1) GO TO 8891
      DO 8889 MMM=1,NPOINT
 8889 WRITE(32,8890) MMM,X(MMM),Y(MMM)
 8890 FORMAT(1X,I6,2X,2F10.2)
 8891 CONTINUE
      CSPAN=(XMAX-XMIN)/NCOL
      RSPAN=(YMAX-YMIN)/NROW
      DO 602 I=1,NROW
      DO 601 J=1,NCOL
 601  M(I,J)=0
 602  CONTINUE
C.........BEGIN LOOP TO DETERMINE WHICH CELLS THE PATH GOES THRU--
C............OPERATES ON 2 CONSECUTIVE POINTS(TAPPED IN) AT A TIME
      DO 800 I=1,NPOINT-1
 5    CONTINUE
      IF (IBUG.EQ.1) WRITE(32,8004)
 8004 FORMAT(' 5:')
      IF (X(I).GT.X(I+1)) GO TO 20
      X1=X(I)
      X2=X(I+1)
      Y1=Y(I)
      Y2=Y(I+1)
      GO TO 30
 20   X1=X(I+1)
      X2=X(I)
      Y1=Y(I+1)
      Y2=Y(I)
 30   IF (Y2.GE.Y1) BBB=1.00001
      IF (Y2.LT.Y1) BBB=.99999
      ROW1 = (Y1-YMIN) / RSPAN + BBB
```

Figure 1 - continued

```
      COL1 = (X1-XMIN) / CSPAN + 1.00001
      ROW2 = (Y2-YMIN) / RSPAN + BBB
      COL2 = (X2-XMIN) / CSPAN + 1.00001
      IROW1=ROW1
      ICOL1=COL1
      IROW2=ROW2
      ICOL2=COL2
      IF (IROW1.LT.1) IROW1=1
      IF (ICOL1.LT.1) ICOL1=1
      IF (IROW2.GT.NROW) IROW2=NROW
      IF (ICOL2.GT.NCOL) ICOL2=NCOL
      ROW1=IROW1
      COL1=ICOL1
      ROW2=IROW2
      COL2=ICOL2
      T1 = XMIN + (COL2 - 1.0) * CSPAN
35    CONTINUE
      IF (IBUG.EQ.1) WRITE(32,8003)
8003  FORMAT(' 35:')
      IF (IBUG.EQ.1)
     1  WRITE(32,8001) NROW,X1,ROW1,IROW1,T1,NCOL,X2,COL1,ICOL1,T2,
     1    RSPAN,Y1,ROW2,IROW2,TX1,CSPAN,Y2,COL2,ICOL2,TX2,
     1    EQUATB,BOXX,ROW,IROW,TY1,EQUATM,BOXY,COL,ICOL,TY2,
     1    YCHECK,ADDX,XCROSS,ADDY,YCROSS
      IF (T1.NE.X2) GO TO 40
      ICOL2=ICOL2 - 1
      COL2 = ICOL2
40    IF (Y2.LT.Y1) GO TO 45
      T2 = YMIN + (ROW2 - 1.0) * RSPAN
      IF (T2.NE.Y2) GO TO 50
      IROW2 = IROW2-1
      ROW2 = IROW2
      GO TO 50
45    T2=YMIN + ROW2 * RSPAN
      IF (T2.NE.Y2) GO TO 50
      IROW2 = IROW2+1
      ROW2 = IROW2
50    CONTINUE
      M(IROW1,ICOL1)=1
      M(IROW2,ICOL2)=1
      IF (IBUG.EQ.1) WRITE(32,8005)
8005  FORMAT(' 50:')
      IF (IBUG.EQ.1)
     1  WRITE(32,8001) NROW,X1,ROW1,IROW1,T1,NCOL,X2,COL1,ICOL1,T2,
     1    RSPAN,Y1,ROW2,IROW2,TX1,CSPAN,Y2,COL2,ICOL2,TX2,
```

Figure 1 - continued

```
     1    EQUATB,BOXX,ROW,IROW,TY1,EQUATM,BOXY,COL,ICOL,TY2,
     1    YCHECK,ADDX,XCROSS,ADDY,YCROSS
          IF (IROW1.NE.IROW2) GO TO 80
          DO 70 K=ICOL1,ICOL2
70        M(IROW1,K) = 1
          GO TO 800
80        IF (ICOL1.NE.ICOL2) GO TO 100
          IF (IROW2.GE.IROW1) GO TO 85
          IJK = IROW1
          IROW1=IROW2
          IROW2=IJK
85        CONTINUE
          DO 90 K=IROW1,IROW2
90        M(K,ICOL1) = 1
          IF (IBUG.EQ.1) WRITE(32,8006)
8006      FORMAT(' 90:')
          GO TO 800
100       EQUATB = (Y1 * X2 - Y2 * X1) / (X2 - X1)
          EQUATM = (Y1 - EQUATB) / X1
          BOXX = XMIN + (COL1 - 1.0) * CSPAN
          OFFSET = 0.0
          IF (Y2.GE.Y1) OFFSET = 1.0
          BOXY = YMIN + (ROW1 - OFFSET) * RSPAN
150       CONTINUE
          IF (IBUG.EQ.1) WRITE(32,8009)
8009      FORMAT(' 150:')
          IF (IBUG.EQ.1)
     1    WRITE(32,8001) NROW,X1,ROW1,IROW1,T1,NCOL,X2,COL1,ICOL1,T2,
     1      RSPAN,Y1,ROW2,IROW2,TX1,CSPAN,Y2,COL2,ICOL2,TX2,
     1      EQUATB,BOXX,ROW,IROW,TY1,EQUATM,BOXY,COL,ICOL,TY2,
     1      YCHECK,ADDX,XCROSS,ADDY,YCROSS
          ADDX = BOXX + CSPAN
          IF (Y2.GE.Y1) ADDY = BOXY + RSPAN
          IF (Y2.LT.Y1) ADDY = BOXY - RSPAN
          IF (Y2.LT.Y1) GO TO 120
          IF (X2.GT.ADDX.AND.Y2.GT.ADDY) GO TO 200
          IF (X2.LE.ADDX) GO TO 7778
          KEY2=ICOL+1
          IF (KEY2.GT.ICOL2) GO TO 800
          DO 7777 KEY = ICOL+1,ICOL2
7777      M(IROW2,KEY) = 1
          GO TO 800
7778      IF (Y2.LE.ADDY) GO TO 130
          KEY2=IROW+1
          IF (KEY2.GT.IROW2) GO TO 800
```

Figure 1 - continued

```
        DO 7779 KEY = IROW+1,IROW2
 7779   M(KEY,ICOL2) = 1
        GO TO 800
 120    IF (X2.GT.ADDX.AND.Y2.LT.ADDY) GO TO 200
        IF (X2.LE.ADDX) GO TO 7772
        KEY2=ICOL+1
        IF (KEY2.GT.ICOL2) GO TO 800
        DO 7773 KEY = ICOL+1,ICOL2
 7773   M(IROW2,KEY) = 1
        GO TO 800
 7772   IF (Y2.GE.ADDY) GO TO 130
        KEY2=IROW-1
        IF (KEY2.LT.IROW2) GO TO 800
        DO 7774 KEY=IROW-1,IROW2,-1
 7774   M(KEY,ICOL2) = 1
        GO TO 800
C.........SEGMENT ENDS IN THIS BOX
 130    CONTINUE
        IF (IBUG.EQ.1) WRITE(32,8007)
 8007   FORMAT(' 130:')
        IF (IBUG.EQ.1)
     1  WRITE(32,8001) NROW,X1,ROW1,IROW1,T1,NCOL,X2,COL1,ICOL1,T2,
     1    RSPAN,Y1,ROW2,IROW2,TX1,CSPAN,Y2,COL2,ICOL2,TX2,
     1    EQUATB,BOXX,ROW,IROW,TY1,EQUATM,BOXY,COL,ICOL,TY2,
     1    YCHECK,ADDX,XCROSS,ADDY,YCROSS
        ROW = (BOXY - YMIN) / RSPAN + OFFSET
        COL = (BOXX - SMIN) / CSPAN + 1.0
        IROW = ROW + .00001
        ICOL = COL + .00001
        M(IROW,ICOL) = 1
        GO TO 800
 200    YCHECK = EQUATM * ADDX + EQUATB
 201    CONTINUE
        IF (IBUG.EQ.1) WRITE(32,8008)
 8008   FORMAT(' 201:')
        IF (IBUG.EQ.1)
     1  WRITE(32,8001) NROW,X1,ROW1,IROW1,T1,NCOL,X2,COL1,ICOL1,T2,
     1    RSPAN,Y1,ROW2,IROW2,TX1,CSPAN,Y2,COL2,ICOL2,TX2,
     1    EQUATB,BOXX,ROW,IROW,TY1,EQUATM,BOXY,COL,ICOL,TY2,
     1    YCHECK,ADDX,XCROSS,ADDY,YCROSS
        Z=ABS(YCHECK - ADDY)
        IF (Z.GT.0.00001) GO TO 250
C.........GOES THRU DIAGONAL BOX
        BOXX = ADDX
        BOXY = ADDY
```

Figure 1 - continued

```
        ROW = (BOXY - YMIN) / RSPAN + OFFSET
        COL = (BOXX - XMIN) / CSPAN + 1.0
        IROW = ROW + .00001
        ICOL = COL + .00001
        M(IROW,ICOL) = 1
        GO TO 150
C.........CHECK BOX ABOVE(BELOW)
 250    CONTINUE
        IF (IBUG.EQ.1) WRITE(32,8010)
 8010   FORMAT(' 250:')
        IF (Y2.LT.Y1) GO TO 260
        XCROSS = (BOXY + RSPAN - EQUATB) / EQUATM
        GO TO 265
 260    XCROSS = (BOXY - RSPAN - EQUATB) / EQUATM
 265    TX1 = BOXX
        TX2 = BOXX + CSPAN
        IF (TX1.LE.XCROSS.AND.TX2.GT.XCROSS) GO TO 270
        GO TO 300
 270    CONTINUE
        IF (IBUG.EQ.1) WRITE(32,8011)
 8011   FORMAT(' 270:')
        IF (IBUG.EQ.1)
     1  WRITE(32,8001) NROW,X1,ROW1,IROW1,T1,NCOL,X2,COL1,ICOL1,T2,
     1    RSPAN,Y1,ROW2,IROW2,TX1,CSPAN,Y2,COL2,ICOL2,TX2,
     1    EQUATB,BOXX,ROW,IROW,TY1,EQUATM,BOXY,COL,ICOL,TY2,
     1    YCHECK,ADDX,XCROSS,ADDY,YCROSS
        IF (Y2.LT.Y1) GO TO 280
        BOXY = BOXY + RSPAN
        GO TO 285
 280    BOXY = BOXY - RSPAN
 285    ROW = (BOXY - YMIN) / RSPAN + OFFSET
        COL = (BOXX - XMIN) / CSPAN + 1.0
        IROW = ROW + .00001
        ICOL = COL + .00001
        M(IROW,ICOL) = 1
        GO TO 150
C.........CHECK BOX TO THE RIGHT
 300    CONTINUE
        IF (IBUG.EQ.1) WRITE(32,8012)
 8012   FORMAT(' 300:')
        IF (IBUG.EQ.1)
     1  WRITE(32,8001) NROW,X1,ROW1,IROW1,T1,NCOL,X2,COL1,ICOL1,T2,
     1    RSPAN,Y1,ROW2,IROW2,TX1,CSPAN,Y2,COL2,ICOL2,TX2,
     1    EQUATB,BOXX,ROW,IROW,TY1,EQUATM,BOXY,COL,ICOL,TY2,
     1    YCHECK,ADDX,XCROSS,ADDY,YCROSS
```

Figure 1 - continued

```
      YCROSS = (BOXX + CSPAN) * EQUATM + EQUATB
      IF (Y2.LT.Y1) GO TO 310
      TY1 = BOXY
      TY2 = BOXY + RSPAN
      IF (TY1.LE.YCROSS.AND.TY2.GT.YCROSS) GO TO 360
      GO TO 340
310   TY1 = BOXY - RSPAN
      TY2 = BOXY
      IF (TY1.LT.YCROSS.AND.TY2.GE.YCROSS) GO TO 360
340   WRITE(5,901)
901   FORMAT(1X,'ERROR')
      DO 350 KK=NROW,1,-1
350   WRITE(33,902) (M(KK,ICOL),ICOL=1,NCOL)
902   FORMAT(1X,200I1)
      GO TO 888
360   CONTINUE
      IF (IBUG.EQ.1) WRITE(32,8013)
8013  FORMAT(' 360:')
      IF (IBUG.EQ.1)
     1  WRITE(32,8001) NROW,X1,ROW1,IROW1,T1,NCOL,X2,COL1,ICOL1,T2,
     1    RSPAN,Y1,ROW2,IROW2,TX1,CSPAN,Y2,COL2,ICOL2,TX2,
     1    EQUATB,BOXX,ROW,IROW,TY1,EQUATM,BOXY,COL,ICOL,TY2,
     1    YCHECK,ADDX,XCROSS,ADDY,YCROSS
      BOXX = BOXX + CSPAN
      ROW = (BOXY - YMIN) / RSPAN + OFFSET
      COL = (BOXX - XMIN) / CSPAN + 1.0
      IROW = ROW + .00001
      ICOL = COL + .00001
      M(IROW,ICOL) = 1
      GO TO 150
800   CONTINUE
      DO 72 III=NROW,1,-1
72    WRITE(31,9876) (M(III,J),J=1,NCOL)
9876  FORMAT(200I1)
      NCELL= NCOL * NROW
      ITOT = 0
      DO 8666 MM1=1,NROW
      DO 8667 MM2=1,NCOL
      IF (M(MM1,MM2).EQ.1) ITOT = ITOT + 1
8667  CONTINUE
8666  CONTINUE
      WRITE(5,8669) NCELL,ITOT
8669  FORMAT(1X,'TOTAL NUMBER OF CELLS = ',I6,/,
     1    1X,'TOTAL NUMBER OF CELLS IN THE PATH = ',I6)
      WRITE(5,8998)
```

Figure 1 - continued

```
8998    FORMAT(/,1X,'HOW MANY MAPS DO YOU WISH TO SUPERIMPOSE THIS ',
     1    'PATH ON (MAPS MUST BE IN',/,3X,'FILES FOR61.DAT,FOR62.DAT,',
     2    'FOR63.DAT,...,ETC. IN SYMAP MATRIX FORMAT)? >',$)
        READ(24,3004) NUNITS
        IF (NUNITS.EQ.0) GO TO 888
        NUNITS = NUNITS + 60
        DO 887 IUNIT=61,NUNITS
        TOTR=0.
        REWIND 31
        REWIND IUNIT
        READ(IUNIT,6111) KROW,KCOL
        MROW=MINO(KROW,NROW)
        MCOL=MINO(KCOL,NCOL)
6111    FORMAT(2I5)
        DO 6113 MY=MROW,1,-1
        READ(IUNIT,6112) (RVAL(MY,MX),MX=1,KCOL)
6112    FORMAT(8F12.5)
        DO 6114 MX=1,MCOL
        A = M(MY,MX)
        RVAL(MY,MX) = RVAL(MY,MX) * A
6114    TOTR = TOTR + RVAL(MY,MX)
6113    CONTINUE
        WRITE(5,8670) TOTR
8670    FORMAT(1X,'TOTAL VALUE OF CELLS IN THE PATH = ',F15.4)
        XTOT = ITOT
        AVGR = TOTR / XTOT
        WRITE(5,8671) AVGR
8671    FORMAT(1X,'AVERAGE VALUE OF CELLS IN THE PATH = ',F15.4)
887     CONTINUE
888     CONTINUE
        STOP
8001    FORMAT(' NROW',I10,' X1',F12.3,' ROW1',F10.3,' IROW1',I9,' T1',
     1  F12.3,/,' NCOL',I10,' X2',F12.3,' COL1',F10.3,' ICOL1',I9,' T2',
     1  F12.3,/,' RSPAN',F9.3,' Y1',F12.3,' ROW2',F10.3,' IROW2',
     1  I9,' TX1',F11.3,/,' CSPAN',F9.3,' Y2',F12.3,' COL2',F10.3,
     1  ' ICOL2',I9,' TX2',F11.3,/,' EQUATB',F8.3,' BOXX',F10.3,' ROW',
     1  F11.3,' IROW',I10,' TY1',F11.3,/,' EQUATM',F8.3,' BOXY',F10.3,
     1  ' COL',F11.3,' ICOL',I10,' TY2',F11.3,/,' YCHECK',F8.3,
     1  ' ADDX',F10.3,' XCROSS',F8.3,' ADDY',F10.3,' YCROSS',F8.3,/)
        END
```

Figure 1 - continued

Appendix C - Computer Program PATH

PATH

This routine uses the Dijkstra algorithm to find minimum path costs in directed networks. The Dijkstra method is used as originally described with the following two asumptions:

1) all nodes are arranged and ordered in an M by N matrix, and
2) all branches are predefined vectors. Each node has identical incoming and outgoing vectors (eight). Each branch has a magnitude defined in array "weight" and a direction of one of the eight adjacent (neighboring) nodes.

In order for both conditions to be satisfied a new set of nodes must be defined. Let a node be a member of set D if:

1) the node does not belong to set A or set B,
2) the node has fewer than 8 adjacent nodes (i.e. edge members of the cost matrix), or
3) the node is of undefined or infinite cost.

Set D nodes are never considered in the network search and therefore may never be added to set A.

These modifications have been introduced to allow for constraints placed on the user by limited machine resources (e.g. core and processor time). By imposing use of a matrix and uniform vectors (branches) both core requirements and CP time are significantly reduced.

One problem encountered using the matrix and defined branch method is cylindrical wrap-around introduced when using indexes to define vector directions. For instance, when referring to a node one to left in the same

row, simply subtract one from the current node. Unfortunately this falls down for all nodes in column one since one less will point to the last node in the previous row. To avoid this problem the shell matrix technique is used. An M X N matrix is placed in the middle of an (M + 2) by (N + 2) matrix, subsequently, all edge nodes are placed in set D (as undefined) and are never searched, thus eliminating the wrap-around problem. An added benefit of the new set D is that it may be used for other than edge nodes. If data is unavailable for a particular node (as is common in the real world) placing it in set D (a negative value) will cause it to be ignored.

Calling Instructions

Requires: Matrix - cost matrix in original row-column form
NROWS - number of rows in original cost matrix
NCOLS - number of columns in original cost matrix
NODE1 - ordinal position of starting node in the original cost matrix
NODEE - ordinal position of ending node in the original cost matrix (zero for all)

Returns: Matrix - cost matrix in shell form
MINCST - as documented under 'Arrays'
INFO - as documented under 'Arrays'
LASTND - as documented under 'Arrays'

Arrays

MATRIX - A shell matrix that contains the NROWS * NCOLS cost matrix surrounded by a one layer shell of nodes of infinite cost. Size: (NROWS + 2) * (NCOLS + 2).

MINCST - An array to store respective temporary and final minimum path costs from the starting node. All values should be initialized large. Size: NROWS * NCOLS.

INFO - Flags to identify each node. If node belongs to:

Set A - Negative value

Set B - Positive value

Set C - Zero

Set D - Zero

Values for set B nodes are pointers in a 'from-to' linked list. All set B nodes are ordered by cost from lowest to highest to enable ease of selection of the next set A node and to eliminate redundant searches for the lowest set B node. The format of the word is: from * ISCAL + to. Where 'from' represents the node of lower value and 'to' the node of higher value (ISCAL was chosen as a scaling factor since DEC-10 FORTRAN optimizes division by a power of 2 constant into an arithmetic shift). The first NCELLS flags should be set to zero, thereby placing all nodes in set C or set D. The last word, a dummy word to anchor the set B linked list, should be set to 999 indicating this is the final node. Size: NROWS * NCOLS + 1.

LASTND - An array containing 'from' linked list information. All set A and set B nodes point back one node along a minimum cost path. By following consecutive nodes back to the starting node one can determine the entire path. The pointer from the starting node should be a large number and all values should be initialized to zero (i.e. set C). Size: NROWS * NCOLS + 1.

NGHBRS - An array of indexes that represent set II branch directions from any given node, specifically, these represent nearest 'neighbors' to a node. For instance, given node I in a shell matrix, I + NGHBRS (2) would point to the node directly above I and I + NGHBRS (8) would point one node down and to the right. Indexes start at the upper-left diagonal and continue clockwise (if only orthogonal paths are desired, diagonal paths may be eliminated by deleting members 1, 3, 6, and 8 of the NGHBRS array and shrinking its size to 4). Size: 8.

WEIGHT - Sister array of NGHBRS. Array NGHBRS and WEIGHT jointly define set II branches. WEIGHT is used to specify magnitude of each vector. Although currently all are set to unity, the user is urged to consider setting diagonal vectors to SQRT (2). Size: (8).

SCALARS

NROWS - number of rows in original cost matrix

NCOLS - number of columns in original cost matrix

NCELLS - total number of cells in original cost matrix

NSROWS - number of rows in shell matrix

NSCOLS - number of columns in shell matrix

NSCELL - total number of cells in shell matrix

NODE1 - starting node as defined by the calling program

NODEE - ending node as defined by the calling program (all nodes if zero)

INFINT - a number larger than all expected path costs

LASTA - pointer to the node of set A currently in use

NEWSND - shell matrix subscript for current nearest neighbor under consideration

NEWNOD - original matrix subscript for current nearest neighbor under consideration

NEWCST - temporary storage for cost to current nearest neighbor under consideration

LSTANC - pointer to the anchor word of the set B linked list (always the extra word in info)

LSTPTR - pointer to the top node (lowest cost) in the set B linked list

LINKFR - used as temporary storage when breaking apart a member of the set B linked list. Holds 'from' pointer.

LINKTO - used as temporary storage when breaking apart a member of the set B linked list. Holds 'to' pointer.

ISCAL - scaling factor used to pack both 'to' and 'from' pointers of the set B linked list into the same word

NEWPTR - temporary storage for holding pointers while searching the set B linked list

LISTND - temporary storage for holding pointers while searching the set B linked list

Statement Functions

NXSXR - node conversion from shell subscript to real subscript

NXRXS - node conversion from real subscript to shell subscript

Since the original cost matrix is enclosed within a shell matrix it is often necessary to know the ordinal location of a node in both matrices (array matrix will always use a 'shell' subscript, while arrays MINCST, INFO, LASTND will always use a 'real' subscript). Therefore two statement functions have been provided for easy conversion. Note: nodes are ordered left to right starting in the upper-left corner.

```
C       THIS IS THE PROGRAM PATH FOR COMPUTATION OF
C       OPTIMUM LOCATION FOR A PATH THROUGH A MATRIX
      DIMENSION MATRIX(20436),INFO(19867),MINCST(19866),LASTND(19867)
      DIMENSION TEMP(16),FRMT(16)
      REAL MATRIX,MINCST
      INTEGER ANS,UNIT,START,END,TEMP,OUT
      DATA IN,OUT,UNIT/5,5,23/,PATH,CORR/2*.FALSE./
      NXRXS(J)=J+(2*((J-1)/NCOLS)+NSCOLS+1)
      WRITE(OUT,1)
1     FORMAT(///' DEFINE MATRIX SIZE'//' NUMBER OF ROWS? '3X$)
      READ(IN,2)NROWS
2     FORMAT(2I)
      WRITE(OUT,3)
3     FORMAT(' NUMBER OF COLUMNS? '$)
      READ(IN,2)NCOLS
      NCELLS=NROWS*NCOLS
      WRITE(OUT,200)
200   FORMAT(/' MINIMUM VALUE? '$)
      READ(IN,201)XMIN
201   FORMAT(F)
      WRITE(OUT,202)
202   FORMAT(/' MAXIMUM VALUE? '$)
      READ(IN,201)XMAX
      WRITE(OUT,4)
4     FORMAT(/' STARTING NODE? '$)
      READ(IN,2)NODE1,NODET
      IF(NODET.NE.0)NODE1=(NODE1-1)*NCOLS+NODET
      WRITE(OUT,5)
5     FORMAT(' ENDING NODE (ZERO OR RETURN FOR ALL)? '$)
      READ(IN,2)NODEE,NODET
      IF(NODET.NE.0)NODEE=(NODEE-1)*NCOLS+NODET
      WRITE(OUT,6)UNIT
6     FORMAT(/' IS MATRIX ON UNIT ',I2,'? '$)
      READ(IN,7)ANS
7     FORMAT(A1)
      IF(ANS.EQ.1HY)GO TO 9
      WRITE(OUT,8)
8     FORMAT(' SPECIFY UNIT NUMBER: '$)
      READ(IN,2)UNIT
9     WRITE(OUT,10)
10    FORMAT(' SPECIFY MATRIX FORMAT:')
      READ(IN,11)FRMT
11    FORMAT(16A5)
      WRITE(OUT,12)
12    FORMAT(//' THANK YOU'////)
```

Figure 1 - Computer Program Listing of PATH

```
      DO 13 I=1,NROWS
13    READ(UNIT,FRMT)(MATRIX((I-1)*NCOLS+J),J=1,NCOLS)
      DO 46 I=1,NCELLS
      IF(MATRIX(I).EQ.0)MATRIX(I)=-1
C     IF(MATRIX(I).GT.0)MATRIX(I)=1.+(MATRIX(I)-XMIN)/(XMAX-XMIN)
46    CONTINUE
47      CONTINUE
      CALL PATHS(MATRIX,MINCST,INFO,LASTND,NROWS,NCOLS,NODE1,NODEE)
      NDSAVE=NODEE
      WRITE(OUT,50)
50    FORMAT('0WOULD YOU LIKE A LISTING OF THE PATH? (YES OR NO) '$)
      READ(IN,7)ANS
      IF(ANS.NE.1HY)GO TO 75
      IF(NODEE.NE.0)GO TO 65
      WRITE(OUT,55)
55    FORMAT(' TO WHICH POINT? (RETURN FOR ALL) '$)
60    READ(IN,2)NODEE,NODET
      IF(NODET.NE.0)NODEE=(NODEE-1)*NCOLS+NODET
65    CALL LIST(LASTND,MINCST,NODE1,NODEE,NCOLS,NCELLS)
      IF(NODEE.EQ.0.OR.NDSAVE.NE.0)GO TO 75
      WRITE(OUT,70)
70    FORMAT(///' ANOTHER POINT? (YES OR NO) '$)
      READ(IN,7)ANS
      IF(ANS.EQ.1HY)GO TO 60
75    NODEE=NDSAVE
77    WRITE(OUT,80)
80    FORMAT(///' WOULD YOU LIKE A PATH MAP? (YES OR NO) '$)
      READ(IN,7)ANS
      IF(ANS.EQ.1HY)PATH=.TRUE.
      WRITE(OUT,85)
85    FORMAT(' WOULD YOU LIKE A CORRIDOR MAP? (YES OR NO) '$)
      READ(IN,7)ANS
      IF(ANS.EQ.1HY)CORR=.TRUE.
      IF(.NOT.PATH.AND..NOT.CORR)GO TO 115
      IF(NODEE.NE.0)GO TO 100
90    WRITE(OUT,95)
95    FORMAT(' ENDING POINT? '$)
      READ(IN,2)NODEE,NODET
      IF(NODET.NE.0)NODEE=(NODEE-1)*NCOLS+NODET
100   CALL PRESET(MATRIX,LASTND,NROWS,NCOLS,NODEE,NODE1)
      IF(PATH)CALL MAP(MATRIX,NROWS,NCOLS,1)
      IF(.NOT.CORR)GO TO 105
      CALL CORRDR(MATRIX,NROWS,NCOLS)
      CALL MAP(MATRIX,NROWS,NCOLS,2)
105   IF(NDSAVE.NE.0)GO TO 115
```

Figure 1 - continued

```
      WRITE(OUT,110)
110   FORMAT(' ANOTHER MAP? '$)
      READ(IN,7)ANS
      IF(ANS.NE.1HY)GO TO 115
      PATH=.FALSE.
      CORR=.FALSE.
      GO TO 77
115   CONTINUE
      STOP
      END
      SUBROUTINE PATHS(MATRIX,MINCST,INFO,LASTND,NROWS,NCOLS,NODE1,NODEE
     $)
C
C
C                              *** PATHS ***
C
C
      DIMENSION MATRIX(1),MINCST(1),INFO(1),LASTND(1)
      DIMENSION NGHBRS(8),WEIGHT(8)
      REAL MATRIX,MINCST,NEWCST
C
C                               *** SCALARS ***
C
C
C     NROWS  - NUMBER OF ROWS IN ORIGINAL COST MATRIX.
C     NCOLS  - NUMBER OF COLUMNS IN ORIGINAL COST MATRIX.
C     NCELLS - TOTAL NUMBER OF CELLS IN ORIGINAL COST MATRIX.
C     NSROWS - NUMBER OF ROWS IN SHELL MATRIX.
C     NSCOLS - NUMBER OF COLUMNS IN SHELL MATRIX.
C     NSCELL - TOTAL NUMBER OF CELLS IN SHELL MATRIX.
C     NODE1  - STARTING NODE AS DEFINED BY THE CALLING PROGRAM.
C     NODEE  - ENDING NODE AS DEFINED BY THE CALLING PROGRAM
C                    (ALL NODES IF ZERO).
C     INFINT - A NUMBER LARGER THAN ALL EXPECTED PATH COSTS.
C
C     LASTA  - POINTER TO THE NODE OF SET A CURRENTLY IN USE.
C     NEWSND - SHELL MATRIX SUBSCRIPT FOR CURRENT NEAREST NEIGHBOR UNDER
C                   CONSIDERATION.
C     NEWNOD - ORIGINAL MATRIX SUBSCRIPT FOR CURRENT NEAREST NEIGHBOR
C                   UNDER CONSIDERATION.
C     NEWCST - TEMPORARY STORAGE FOR COST TO CURRENT NEAREST NEIGHBOR
C                   UNDER CONSIDERATION.
C
C     LSTANC - POINTER TO THE ANCHOR WORD OF THE SET B LINKED LIST
C                   (ALWAYS THE EXTRA WORD IN INFO).
```

Figure 1 - continued

```
C     LSTPTR - POINTER TO THE TOP NODE (LOWEST COST) IN THE SET B
C                   LINKED LIST.
C     LINKFR - USED AS TEMPORARY STORAGE WHEN BREAKING APART A MEMBER
C                   OF THE SET B LINKED LIST. HOLDS 'FROM' POINTER.
C     LINKTO - USED AS TEMPORARY STORAGE WHEN BREAKING APART A MEMBER
C                   OF THE SET B LINKED LIST. HOLDS 'TO' POINTER.
C     ISCAL  - SCALING FACTOR USED TO PACK BOTH 'TO' AND 'FROM' POINTERS
C                   OF THE SET B LINKED LIST INTO THE SAME WORD.
C     NEWPTR - TEMPORARY STORAGE FOR HOLDING POINTERS WHILE SEACHING
C                   THE SET B LINKED LIST.
C     LISTND - TEMPORARY STORAGE FOR HOLDING POINTERS WHILE SEACHING
C                   THE SET B LINKED LIST.
C
      DATA INFINT/999999999/,ISCAL/32768/,NNBRS/8/
C
C
C
C
C                *** STATEMENT FUNCTIONS ***
C
C
      NXSXR(J)=J-(2*((J-1)/NSCOLS)+NSCOLS-1)
      NXRXS(J)=J+(2*((J-1)/NCOLS)+NSCOLS+1)
C
C
C     NXSXR - NODE CONVERSION FROM SHELL SUBSCRIPT TO REAL SUBSCRIPT
C     NXRXS - NODE CONVERSION FROM REAL SUBSCRIPT TO SHELL SUBSCRIPT
C
C     SINCE THE ORIGINAL COST MATRIX IS ENCLOSED WITHIN A SHELL MATRIX
C     IT IS OFTEN NECESSARY TO KNOW THE ORDINAL LOCATION OF A NODE IN
C     BOTH MATRICES (ARRAY MATRIX WILL ALWAYS USE A 'SHELL' SUBSCRIPT,
C     WHILE ARRAYS MINCST, INFO, LASTND WILL ALWAYS USE A 'REAL'
C     SUBSCRIPT). THEREFORE TWO STATEMENT FUNCTIONS HAVE BEEN PROVIDED
C     FOR EASY CONVERSION. NOTE: NODES ARE ORDERED LEFT TO RIGHT START-
C     ING IN THE UPPER-LEFT CORNER.
C
C
C                  *** INITIALIZATION ***
C
C
      NCELLS=NROWS*NCOLS
      NSROWS=NROWS+2
      NSCOLS=NCOLS+2
      NSCELL=NSROWS*NSCOLS
C
C     MOVE ORIGINAL COST MATRIX NODES INTO SHELL MATRIX LOCATIONS
```

Figure 1 - continued

```
C
      I=NCELLS
10001 MATRIX(NXRXS(I))=MATRIX(I)
      I=I-1
      IF(I)10002,10002,10001
C
C     SET EDGE NODES OF SHELL MATRIX TO INFINITY
C
10002 DO 10003 I=1,NSCOLS-1
10003 MATRIX(I)=-1
      DO 10004 I=NSCOLS,NSCELL-NSCOLS,NSCOLS
      MATRIX(I)=-1
10004 MATRIX(I+1)=-1
      DO 10005 I=NSCELL-NSCOLS+2,NSCELL
10005 MATRIX(I)=-1
C
C     INITIALIZE MINIMUM COSTS, FLAGS AND ROUTING LIST
C
      DO 10006 I=1,NCELLS
      MINCST(I)=INFINT
      INFO(I)=0
10006 LASTND(I)=0
C
C     INITIALIZE SET B LINKED LIST
C
      INFO(NCELLS+1)=999
      LSTANC=NCELLS+1
      LSTPTR=LSTANC
C
C     DEFINE SET II BRANCH VECTORS
C
      NGHBRS(1)=-(NSCOLS+1)
      NGHBRS(2)=-NSCOLS
      NGHBRS(3)=-(NSCOLS-1)
      NGHBRS(4)=-1
      NGHBRS(5)=1
      NGHBRS(6)=NSCOLS-1
      NGHBRS(7)=NSCOLS
      NGHBRS(8)=NSCOLS+1
      WEIGHT(1)=SQRT(1+.8**2)
      WEIGHT(2)=0.8
      WEIGHT(3)=SQRT(1+.8**2)
      WEIGHT(4)=1.0
      WEIGHT(5)=1.0
      WEIGHT(6)=SQRT(1+.8**2)
```

Figure 1 - continued

```
      WEIGHT(7)=0.8
      WEIGHT(8)=SQRT(1+.8**2)
C
C
C         *** BEGIN MODIFIED DIJKSTRA ALGORITM ***
C
C     PLACE STARTING NODE IN SET A
C     AND FLAG AS ANCHOR FOR THE PATHS LINKED LIST
C
      INFO(NODE1)=-1
      LASTND(NODE1)=INFINT
      LASTA=NODE1
C     SET COST FOR STARTING NODE (DIJKSTRA ASSUMES ZERO)
      MINCST(NODE1)=0
C
C
C              *** BEGIN STEP 1 ***
C
C
C     DETERMINE COSTS TO ALL NON-INFINITE SET B AND SET C NODES WHICH
C     ARE NEAREST NEIGHBORS TO THE CURRENT SET A NODE. PLACE ALL SET C
C     NODES IN SET B BY ADDING THEM TO THE LINKED LIST AND ENTERING THE
C     NEW COST IN THE MINCST ARRAY. ALL SET B NODES SHOULD COMPARE NEW
C     COST WITH PREVIOUS MINIMUM PATH COST. IF LOWER, MINCST SHOULD BE
C     UPDATED AND THE NODE DELETED AND READDED TO THE LINKED LIST.
C
1     DO 8 K=1,NNBRS
      NEWSND=NXRXS(LASTA)+NGHBRS(K)
      NEWNOD=NXSXR(NEWSND)
C
C     SKIP THIS ADJACENT NODE IF MEMBER OF SET D
C
        IF(MATRIX(NEWSND))8,2,2
C
C     SKIP THIS NEAREST NEIGHBOR IF ALREADY SET A
C
2     IF(INFO(NEWNOD))8,3,3
C
C     FIND COST AND UPDATE MINCST IF LOWEST ENCOUNTERED THUS FAR.
C     SET C MEMBERS ALWAYS UPDATED SINCE COSTS INITIALIZED AT INFINITY.
C
3     NEWCST=MATRIX(NEWSND)*WEIGHT(K)+MINCST(LASTA)
      IF(NEWCST.GE.MINCST(NEWNOD))GO TO 8
      MINCST(NEWNOD)=NEWCST
C
```

Figure 1 - continued

```
C       DELETE SET B NODE FROM LINKED LIST
C
        IF(INFO(NEWNOD).LE.0)GO TO 4
        LINKFR=INFO(NEWNOD)/ISCAL
        LINKTO=INFO(NEWNOD)-(INFO(NEWNOD)/ISCAL)*ISCAL
        INFO(LINKFR)=(INFO(LINKFR)/ISCAL)*ISCAL+LINKTO
        INFO(LINKTO)=INFO(LINKTO)-(INFO(LINKTO)/ISCAL)*ISCAL+LINKFR*ISCAL
C
C       ADD NODE TO LINKED LIST
C
4       NEWPTR=LSTANC
5       LISTND=NEWPTR
        NEWPTR=INFO(LISTND)/ISCAL
        IF(NEWPTR.EQ.0)GO TO 6
        IF(MINCST(NEWPTR).GT.MINCST(NEWNOD))GO TO 5
C
C       ADD NODE TO MIDDLE OF LIST
C
        INFO(NEWNOD)=NEWPTR*ISCAL+LISTND
        INFO(NEWPTR)=(INFO(NEWPTR)/ISCAL)*ISCAL+NEWNOD
        INFO(LISTND)=INFO(LISTND)-(INFO(LISTND)/ISCAL)*ISCAL+NEWNOD*ISCAL
        GO TO 7
C
C       ADD NODE TO TOP OF LIST
C
6       INFO(LISTND)=INFO(LISTND)+NEWNOD*ISCAL
        INFO(NEWNOD)=LISTND
        LSTPTR=NEWNOD
C
C       PLACE CURRENT BRANCH IN SET II
C       (RECORD THE ROUTE BY STORING THE NODE WE CAME FROM)
C
7       LASTND(NEWNOD)=LASTA
8       CONTINUE
C
C
C               *** BEGIN STEP 2 ***
C
C
C       PLACE THE SET B NODE OF MINIMUM COST (AT THE TOP OF THE LINKED
C       LIST) IN SET A. USE THIS NODE IN STEP 1 PATH EXPANSION.
C
        IF(LSTPTR.EQ.LSTANC)RETURN
C
C       DELETE NODE FROM SET B LINKED LIST AND PLACE IN SET A.
```

Figure 1 - continued

```
C
      NEXTND=INFO(LSTPTR)
      INFO(NEXTND)=INFO(NEXTND)-(INFO(NEXTND)/ISCAL)*ISCAL
      INFO(LSTPTR)=-1
      LASTA=LSTPTR
      LSTPTR=NEXTND
C
C     REPEAT STEPS 1 AND 2 UNTIL ALL ATTAINABLE POINTS
C     ARE IN SET A (IE  LSTPTR = LSTANC).
C
      IF(LASTA.EQ.NODEE)RETURN
      GO TO 1
      END
      SUBROUTINE LIST(LASTND,MINCST,NODE1,NODEE,NCOLS,NCELLS)
      DIMENSION LASTND(1),MINCST(1),TEMP(16)
      REAL MINCST
      INTEGER OUT,START,END,TEMP
      DATA IN,OUT/5,5/
      WRITE(OUT,47)
47    FORMAT(1X,87(1H*))
      DO 48 I=1,NCELLS
      IF(LASTND(I).EQ.0)LASTND(I)=NODE1
48    CONTINUE
      START=NODEE
      END=NODEE
      IF(NODEE)15,14,15
14    START=1
      END=NCELLS
15    DO 22 I=START,END
      N1=(NODE1-1)/NCOLS+1
      N2=NODE1-(N1-1)*NCOLS
      N3=(NODEE-1)/NCOLS+1
      N4=NODEE-(N3-1)*NCOLS
      WRITE(OUT,16)N1,N2,N3,N4,MINCST(I)
16    FORMAT(' STARTING NODE: [',I3,1H,,I3,1H],4X,
     $'ENDING NODE: [',I3,1H,,I3,1H],4X,
     $'MINIMUM PATH COST ',F15.5//' RETURN PATH FROM ENDING NODE TO START
     $ING NODE:')
      LASTND(NCELLS+1)=I
      LAST=NCELLS+1
17    DO 18 J=1,15,2
      TEMP(J)=(LASTND(LAST)-1)/NCOLS+1
      TEMP(J+1)=LASTND(LAST)-(TEMP(J)-1)*NCOLS
      LAST=LASTND(LAST)
      IF(LAST.EQ.NODE1)GO TO 20
```

Figure 1 - continued

```
18    CONTINUE
      WRITE(OUT,19)TEMP
19    FORMAT(8(1X,1HC,I3,1H,,I3,1H]))
      GO TO 17
20    WRITE(OUT,21)(TEMP(K),K=1,J+1)
21    FORMAT(8(1X,1HC,I3,1H,,I3,1H]))
22    WRITE(OUT,23)
23    FORMAT(/1X,87(1H*))
      RETURN
      END
      SUBROUTINE PRESET(MATRIX,LASTND,NROWS,NCOLS,NODEE,NODE1)
      REAL MATRIX
      INTEGER START,END,TEMP
      DIMENSION MATRIX(1),LASTND(1)
      NSCOLS=NCOLS+2
      NCELLS=NROWS*NCOLS
C
C     SET UP MATRIX AGAIN
C         .LE. 0 - NO INFORMATION ON CELL
C         XMIN   - INFORMATION BUT OFF PATH
C         XMAX   - ON PATH
C
26    XMIN=1.0
      XMAX=200.0
      DO 27 I=1,NCELLS
      MATRIX(I)=MATRIX(I+(2*((I-1)/NCOLS)+NSCOLS+1))
      IF(MATRIX(I).GT.0)MATRIX(I)=XMIN
27    CONTINUE
      LASTND(NCELLS+1)=NODEE
      LAST=NCELLS+1
      DO 28 J=1,NCELLS
      MATRIX(LASTND(LAST))=XMAX
      LAST=LASTND(LAST)
      IF(LAST.EQ.NODE1)GO TO 29
28    CONTINUE
29      RETURN
      END
      SUBROUTINE MAP(MATRIX,NROWS,NCOLS,II)
      REAL MATRIX(1)
      INTEGER END,START
      XMIN=1.0
      XMAX=200.0
      I=1
      J=2
      IF(II.NE.1)J=3
```

Figure 1 - continued

```
      K=3
      XXX=31496.
      WRITE(22,34)I,J,K,XXX
34    FORMAT(2I2,I1,F10.2)
      WRITE(22,35)
35    FORMAT(3HEND)
29    WRITE(24)NROWS,NCOLS,X,X,XMIN,XMAX
      START=1
      END=NCOLS
30    IF(END.LE.130)GO TO 32
      DO 31 I=1,NROWS
      NNNN=START+129
31    WRITE(24)(MATRIX((I-1)*NCOLS+J),J=START,NNNN)
      END=END-130
      START=START+130
      GO TO 30
32    DO 33 I=1,NROWS
      NNNN=START+END-1
33    WRITE(24)(MATRIX((I-1)*NCOLS+J),J=START,NNNN)
      RETURN
      END

      SUBROUTINE CORRDR(MATRIX,NROWS,NCOLS)
      REAL MATRIX(1)
      LOGICAL LEFT,RIGHT,TOP,BOTTOM
      DATA XMIN,XMID,XMAX/1.0,100.0,200.0/
      SUBSCR(I,J)=(I-1)*NCOLS+J
      NCELLS=NROWS*NCOLS
C
C     NORMALIZE MATRIX (FILL IN HOLES)
C
      DO 50 J=1,NCOLS
C     FIND LAST ROW WITH INFORMATION IN THIS COLUMN
      K=NROWS+1
10    K=K-1
      IF(K.LT.1)GO TO 50
      IF(MATRIX(SUBSCR(K,J)).LT.XMIN)GO TO 10
C     FIND FIRST ROW WITH INFORMATION IN THIS COLUMN
      DO 20 L=1,K
      IF(MATRIX(SUBSCR(L,J)).GE.XMIN)GO TO 30
20    CONTINUE
      GO TO 50
C     FILL HOLES
30    DO 40 I=L+1,K-1
      IF(MATRIX(SUBSCR(I,J)).LT.XMIN)MATRIX(SUBSCR(I,J))=XMIN
```

Figure 1 - continued

```
40      CONTINUE
50      CONTINUE
C
C       FILL IN NORMAL CORRIDOR POINTS
C
        DO 60 IX=1,NROWS
        DO 60 IY=1,NCOLS
        IF(MATRIX(SUBSCR(IX,IY)).NE.XMAX)GO TO 60
        DO 60 I=1,11
        M=IX-11+I
        IF(M.LT.1.OR.M.GT.NROWS)GO TO 60
        DO 60 J=1,11
        N=IY-11+J
        IF(N.LT.1.OR.N.GT.NCOLS)GO TO 60
        IF(MATRIX(SUBSCR(M,N)).GT.0)MATRIX(SUBSCR(M,N))=XMID
60      CONTINUE
C
C       EXTEND EDGES LEFT-RIGHT IF NEEDED
C
        DO 140 I=1,NROWS
        LEFT=.TRUE.
        RIGHT=.FALSE.
        IRGHT=0
        LFT=0
        DO 90 J=1,NCOLS
        IF(MATRIX(SUBSCR(I,J)).LT.XMID)GO TO 80
        IF(MATRIX(SUBSCR(I,J)).EQ.XMAX)GO TO 70
        IF(RIGHT)IRGHT=IRGHT+1
        IF(LEFT)LFT=LFT+1
        GO TO 90
70      RIGHT=.TRUE.
        LEFT=.FALSE.
        MIDPNT=J
        GO TO 90
80      IF(.NOT.RIGHT)GO TO 100
90      CONTINUE
        GO TO 140
100     IF(((IRGHT.LT.5).AND.(LFT.LT.5)).OR.
     $          ((IRGHT.GE.5).AND.(LFT.GE.5)))GO TO 140
        NXTEND=10-(IRGHT+LFT)
        IF(IRGHT.LT.5)GO TO 120
        DO 110 M=1,NXTEND
        IF(MIDPNT+IRGHT+M.GT.NCOLS)GO TO 110
        N=SUBSCR(I,MIDPNT+IRGHT+M)
        IF(MATRIX(N).LT.XMIN)GO TO 140
```

Figure 1 - continued

```
          MATRIX(N)=XMID
110       CONTINUE
          GO TO 140
120       DO 130 M=1,NXTEND
          IF(MIDPNT-LFT-M.LT.1)GO TO 130
          N=SUBSCR(I,MIDPNT-LFT-M)
          IF(MATRIX(N).LT.XMIN)GO TO 140
          MATRIX(N)=XMID
130       CONTINUE
140       CONTINUE
C
C         EXTEND EDGES TOP-BOTTOM IF NEEDED
C
          DO 220 J=1,NCOLS
          TOP=.TRUE.
          BOTTOM=.FALSE.
          ITP=0
          IBTTM=0
          DO 170 I=1,NROWS
          IF(MATRIX(SUBSCR(I,J)).LT.XMID)GO TO 160
          IF(MATRIX(SUBSCR(I,J)).EQ.XMAX)GO TO 150
          IF(BOTTOM)IBTTM=IBTTM+1
          IF(TOP)ITP=ITP+1
          GO TO 170
150       BOTTOM=.TRUE.
          TOP=.FALSE.
          MIDPNT=I
          GO TO 170
160       IF(.NOT.BOTTOM)GO TO 180
170       CONTINUE
          GO TO 220
180       IF(((IBTTM.LT.5).AND.(ITP.LT.5)).OR.
     $           ((IBTTM.GE.5).AND.(ITP.GE.5)))GO TO 220
          NXTEND=10-(IBTTM+ITP)
          IF(IBTTM.LT.5)GO TO 200
          DO 190 M=1,NXTEND
          IF(MIDPNT+IBTTM+M.GT.NROWS)GO TO 190
          N=SUBSCR(MIDPNT+IBTTM+M,J)
          IF(MATRIX(N).LT.XMIN)GO TO 220
          MATRIX(N)=XMID
190       CONTINUE
          GO TO 220
200       DO 210 M=1,NXTEND
          IF(MIDPNT-ITP-M.LT.1)GO TO 210
          N=SUBSCR(MIDPNT-ITP-M,J)
          IF(MATRIX(N).LT.XMIN)GO TO 220
          MATRIX(N)=XMID
210       CONTINUE
220       CONTINUE
          RETURN
          END
```

Figure 1 - continued

Appendix D - Computer Program TRAIL

TRAIL

TRAIL consists of two subprograms, SLMATH and PSORCA, which were written independently and have been grafted together. SLMATH is a program for forming weighted functions of SYMAP-type matrices, and PSORCA is a program for selecting corridors through a SYMAP matrix. Having them together in a single program, the user is able to composite two maps, then select a corridor through the resulting map in a single run. Of course, SLMATH and PSORCA may still be run separately in the present version of TRAIL.

Changes made to SLMATH in the process of combining the programs will be described. More detailed instructions will be provided for PSORCA.

Input and Output Files

As described in the SLMATH documentation, SLMATH requires input matrices on TAPE 8 and TAPE 11. The output (composite) matrix from subroutine SLMATH in TRAIL is placed on a CDC file named MAP. MAP is then used as the input file for subroutine PSORCA. If the user wishes to skip SLMATH and just run PSORCA, he must place the input matrix on a file named map and not on TAPE 8 or TAPE 11. PSORCA produces an output file named POINTS, which contains row and column coordinates for the points in the selected corridors.

Data Cards

TRAIL requires a number of data cards to function. These are divided into three groups.

Group I - This is a single card which tells TRAIL whether to run SLMATH, PSORCA, or both.

Group II - This group contains all the control cards described in the SLMATH documentation. There are a couple of very minor changes described below. This group of cards should be included if and only if SLMATH has been specified in Group I.

Group III - This group of cards gives directions to PSORCA. It should be included if and only if PSORCA has been specified in Group I.

Details follow. (Note: All integer variables must be right justified in their fields.)

Group	Card	Column	Variable
I	1	5	IMAP
		6-10	NRUN
II	1	1-2	LEVEL
		3	LMAP
		4-13	SCALE
		14	IPT
		15-16	IMP1
		17-18	IMP2
		19-28	WT1
		29-38	WT2
		39	ITRG
		40	FLIP
		41-42	ILEV
	2-7	(See SLMATH documentation)	
	8	This card is for a title and comments	
III	1	Title and comments for the PSORCA run	
	2	1-10	NSPEC
		11-20	W1
		21-30	W2
		31-40	W3
		50	KO
		60	NOB
		70	IMUL
		80	ISL
	3	1-10	MINDX
	(Use this card only if	11-20	MAXDX
	IMUL does not equal zero)	21-30	GUIDE
	4	1-10	MROW
	(Incl. one of these cards	11-20	MCOL
	for each point specified by NSPEC)		

Repeat cards 2-4 in Group III once for each corridor specified by NRUN.

Explanation of Variables

IMAP: Number of matrices to be composited by SLMATH (1 or 2). A zero or blank here will cause SLMATH to be skipped. In this case, the input matrix must be on a file named MAP for PSORCA to use.

NRUN: Number of corridors to be selected by PSORCA through the matrix on file map. A zero or blank here will cause PSORCA to be skipped. If both IMAP and NRUN are zero, the program will terminate without doing a single thing.

See SLMATH documentation for explanations of cards in Group II. Note that IMAP has been placed in Group I, and the locations of the other variables on Card III 1 have been shifted to the left. Also note that only one comment card is allowed for SLMATH, along with one for the entire PSORCA run.

NSPEC: This and all following work must be repeated for each corridor specified by NRUN. NSPEC is the number of points you are going to require a corridor to pass through. There must be at least two such (beginning and end) plus any intermediate points desired. Row and column coordinates for these points will be given later.

W1: This and the next two variables serve the function of guiding PSORCA as it winds its devious way through the matrix. High values of these variables cause the corridor to stay close to a straight line connecting the specified points. Lower values allow the program to deviate from a straight line in order to reach better (i.e. higher-valued) parts of the matrix. Too low values may give rise to loops, figure-eights and other interesting but unfeasible corridors. Zero values will cause PSORCA to ignore its

destination and go where it pleases, usually to a bad end. Each variable does a slightly different job. W1 becomes greater the farther the program has traveled through the matrix. This guarantees that sooner or later the path will be forced to reach the destination point.

W2: This variable becomes greater the closer PSORCA gets to its destination. This guarantees that if the program happens to wander into the right general area, that it will be induced to finish the corridor straight-off, rather than meandering away again.

W3: This weight stays constant through the run, providing a steady tendency to head in the right direction.

These three direction-finding weights are what make PSORCA so much fun. There are infinite combinations of weights which can be tried, of which only a few produce any useful results. The appropriate values for the weights will depend on the nature of the surface represented by the input matrix. A good way to start is probably to give all three weights the same value. That value should approximately equal the greatest amount by which map values change between adjacent cells. For example, if the greatest amount by which adjacent cells differ is ten units of whatever the map represents, set all three weights equal to ten. That should put you in the ball park, and additional adjustments can be made to tighten or loosen the constraints by changing some or all of the weights.

K0: This gives the general direction the corridor is to start out in, and takes values from one to eight. Two is straight up toward the top of the map, four is to the right, six is down and eight is to the left. Odd numbers fall on diagonals. If this is left blank, K0 will take a default value of six.

NOB: The purpose of this switch is to prevent the program from zig-zagging on descents. The switch is on by default when this column is left blank.

IMUL: Setting this equal to one gives rise to a multistage search. PSORCA will search ahead and pick intermediate points in the matrix which are high in value and then go toward those points. IMUL gives the program "foresight" in a sense. If you use IMUL you must include card III 3. This option often gives substantially better corridors, but also adds execution time.

ISL: This switch causes PSORCA to ignore the surface it is on and follow a straight line. This is very useful, since the average value of a straight-line corridor can be used as a standard of comparison for other corridors. Setting ISL to one produces the straight line. This switch will cause W1, W2, and W3 to be disregarded, so you don't need to specify them when ISL is used.

MINDX: This and the next two variables (all on card III 3) are to be used if and only if IMUL was set equal to one. If IMUL is left equal to zero, do not include card III 3. When a multistage search is requested, the program skips checking cell values there.

MAXDX: This is the number of rows ahead at which the program stops checking cell values. Example - with MINDX = 10 and MAXDX = 30, the program examines all cells which are between 10 and 30 rows ahead of the current position in the matrix and picks the one with the highest value. It then treats this as an intermediate point, i.e. it moves toward it. When it gets there, it again looks ahead and repeats the procedure until it winds up at its destination.

GUIDE: This is a direction-finding weight, similar in spirit to W1, W2, and W3, which is used in the multistage search. High values of GUIDE cause the program to favor points lying close to a straight line between the end points of the corridor when it is searching ahead for the next point to go to, low values leave the program free to choose points which are high in value even if they are out of the way. Use the same guidelines for setting the value of GUIDE as were given for W1, W2, and W3.

MROW and MCOL: These are the row and column coordinates of the points specified by NSPEC (remember NSPEC?) for the corridor to go through. Each point gets a separate card; the number of cards here must equal the number given by NSPEC. The program will pass through the points in the order in which you list them.

Cards III 2-4 are repeated over again for each additional corridor you want selected. All the variables and point specifications may be changed for the different corridors.

Printed Output

Printed output from SLMATH is described in the SLMATH documentation.

Output from PSORCA consists of a list of the points included in each selected corridor, giving their row and column coordinates and their map values. Specified points and points selected by a multistage search are identified as such. Following the list PSORCA gives the length of the corridor in cell units and the average value of the corridor.

Finally, PSORCA prints out a map of the corridor. Specified points are shown as "S", multistage search points appear as "+", and other points are given as "*". If the SYMAP matrix has two panels PSORCA will superimpose the two panels of the corridor map. PSORCA cannot handle maps with more than two panels.

Helpful Hints and Words of Warning

There are a couple of pitfalls to avoid when using PSORCA. First, the program is designed for maps containing positive values within the map outline and zero values outside the outline.

Another problem will occur sometimes when you have the direction-finding weights set too low and/or are using a large matrix. Various variables are dimentioned in such a way that corridors may be no more than 500 cells long. If the program does a lot of wandering, the corridor may exceed 500 cells. When this happens, the computer begins writing over variables adjacent to the filled up arrays in core. The cure is to increase the direction-finding weights to stop the program from wandering excessively.

```
      PROGRAM TRAIL(INPUT,OUTPUT,TAPE5=INPUT,TAPE6
     1  =OUTPUT,TAPE8=/200,MAP,TAPE12=MAP,TAPE10,TAPE11,TAPE9
     1  ,TAPE1=MAP,TAPE2,POINTS,TAPE60=POINTS)
C
C
      PRINT 1
 1    FORMAT("1"//////" ***** TRAIL *****"///
     1  " PROGRAM FOR COMPOSITE MAPPING AND CORRIDOR SELECTION")
      READ 5, IMAP,NRUN
 5    FORMAT (2I5)
      IF (IMAP.EQ.0) GO TO 20
      CALL CLMATH(IMAP)
 20   CONTINUE
      IF (NRUN.EQ.0) GO TO 30
      CALL PSORCA(NRUN)
 30   STOP
      END

      SUBROUTINE CLMATH(IMAP)
      INTEGER BLANK,FLIP
      COMMON INP8,INP11,IMP1,IMP2,IPRTCR(10),PTS(130),STS(130),
     1       IPRT(4,131)
      DIMENSION ICHAR(10,4),IP(10,4),ISYM(50),IAREA(50),IL(55),JJ(10)
      DIMENSION RLEV(51),AREA(4,50)
      DATA BLANK,JROW,JCOL/" ",2*80000/
      DATA IP/40*" "/
     DATA IL/1,1,10,1,7,10,1,5,7,10,1,5,7,8,10,1,5,6,8,9,10,1,2,5,6,8,9
     1,10,1,2,5,6,7,8,9,10,1,2,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10/
     DATA IP(1,1),IP(2,1),IP(3,1),IP(4,1),IP(5,1),IP(6,1),IP(7,1)/1H.,
     11H-,1H=,1H+,1H),1HX,1HO/
     DATA IP(8,1),IP(8,2),IP(9,1),IP(9,2),IP(10,1),IP(10,2),IP(10,3),
     1IP(10,4)/"O","-","O","X","O","X","A","V"/
      DATA ISYM/"0","1","2","3","4","5","6","7","8","9","A",
     1       "B","C","D","E","F","G","H","I","J","K","L","M","N","O",
     1        "P","Q","R","S","T","U","V","W","X","Y","Z","*","/","+",
     3        "-","=",":","(",")","\",",","'",".","<",">","$"/
      DATA IAREA/50*0/,AREA/200*0./,IBRD/"I"/
      INP8 = 8
      INP11 = 11
      REWIND 9
      REWIND 10
      REWIND 12
      PRINT 991
```

Figure 1 - Computer Program Listing of TRAIL

```
 991     FORMAT("1"//////" *****SLMATH*****"///
     1   " COMPOSITE MAPPING PROGRAM"//////)
C
        READ(5,10) LEVEL,LMAP,SCALE,IPT,IMP1,IMP2,WT1,WT2,ITRG,
     1              FLIP,ILEV,CONST
10      FORMAT(I2,I1,F10.5,I1,2I2,2F10.5,2I1,I2,F10.2)
        PRINT 992, WT1,WT2,ITRG,CONST
 992    FORMAT(/////" WT1= ",F8.2/" WT2= ",F8.2/" FUNCTION CODE = ",
     1  I2/" CONSTANT = ",F10.2)
        IF(IMP1.NE.0.OR.IMP2.NE.0) CALL READR
C
C    SET SYMBOLS FOR THE BORDERS
C
        DO 5 I=1,10
5       IPRTCR(I) = ISYM(I)
C
C DIFFERENT SYMBOLS
C
```

```
        IF(IPT .GT. 0) READ(5,20) (ISYM(I),I=1,LEVEL)
20      FORMAT(50A1)
C
C BLANK THE PRINT ARRAY
C
        DO 30 I=1,131
         IPRT(1,I) = BLANK
         IPRT(2,I) = BLANK
         IPRT(3,I) = BLANK
30       IPRT(4,I) = BLANK
C
C SET DEFAULT VALUES FOR INPUT
C
        IF(ITRG .LT. 1) ITRG = 1
        IF(WT1  .GT. 0) GO TO 40
        WT1 =  .5
        WT2 = .5
C
40      READ(INP8) IROW,ICOL,DR,DC,VMIN,VMAX
        INPCO = INP8
        IF(IMAP .EQ. 1) GO TO 50
C
```

Figure 1 - continued

```
C INVERT MAP ?
C
        IF(FLIP .GT. 0) GO TO 55
        DUM = VMIN
        VMIN = -VMAX
        VMAX = -DUM
C
55      READ(INP11) JROW,JCOL,CR,CC,VMIN,VMAX
        IF(JROW-IROW + JCOL-ICOL .EQ. 0) GO TO 50
        WRITE(6,1010) JROW,JCOL,IROW,ICOL
1010    FORMAT(1H ,"DATA NOT COMPATBLE"/" PREVIOUS STORED MAP HAS"
     1     ,I5," ROWS AND ",I5," COLUMNS."/" CURRENT MAP HAS",I5,
     2  "  ROWS AND ",I5," COLUMNS")

50      JROW = MINO(IROW,JROW)
        JCOL = MINO(ICOL,JCOL)
        NTIME = 1
        JJ(1) = JCOL
        IF(JCOL .LE. 130) GO TO 70
        NTIME = JCOL / 130
        DO 60 I=1,NTIME
60      JJ(I) = 130
        JJ(NTIME + 1) = JCOL - (NTIME *130)
        IF(NTIME * 130 .LT. JCOL) NTIME = NTIME +1
70      IF(IMAP .EQ. 1) GO TO 210
90      IF(FLIP .GT. 0) WT1 = -WT1
C
C PERFORM DESIRED FUNCTION MANIPULATION ON COMBINED MAPS
C
        DO 200 I=1,NTIME
         J = JJ(I)
         DO 200 K=1,JROW
          READ(INP8)(STS(L),L=1,J)
          READ(INP11)(PTS(L),L=1,J)
100       GO TO (110,130,150,170),ITRG
C
110       DO 120 L=1,J
           PTS(L) = PTS(L) * WT1 + STS(L) * WT2+CONST
        IF(PTS(L).EQ.CONST) PTS(L)=0.




           IF(PTS(L) .LT. VMIN) VMIN = PTS(L)
```

Figure 1 - continued

```
            IF(PTS(L) .GT. VMAX) VMAX = PTS(L)
120         CONTINUE
            GO TO 190
C
130         DO 140 L=1,J
            PTS(L) = PTS(L) * WT1 - STS(L) * WT2+CONST
          IF(PTS(L).EQ.CONST) PTS(L)=0.
            IF(PTS(L) .LT. VMIN) VMIN = PTS(L)
            IF(PTS(L) .GT. VMAX) VMAX = PTS(L)
140         CONTINUE
            GO TO 190
C
150         DO 160 L=1,J
            PTS(L) = PTS(L) * WT1 * STS(L) * WT2+CONST
          IF(PTS(L).EQ.CONST) PTS(L)=0.
            IF(PTS(L) .LT. VMIN) VMIN = PTS(L)

            IF(PTS(L) .GT. VMAX) VMAX = PTS(L)
160         CONTINUE
            GO TO 190
C
170         DO 180 L=1,J
            PTS(L) = (PTS(L) * WT1) / (STS(L) *WT2)+CONST
          IF(PTS(L).EQ.CONST) PTS(L)=0.
            IF(PTS(L) .LT. VMIN) VMIN = PTS(L)
            IF(PTS(L) .GT. VMAX) VMAX = PTS(L)
180         CONTINUE
C
190         WRITE(10) (PTS(L),L=1,J)
200       CONTINUE
          REWIND 10
          INPCO = 10
C
C COMPUTE LEVELS AND THEIR SPACING
C
210       VSUM = (VMAX - VMIN) / LEVEL
          IF(ILEV .LE. 0) GO TO 230
C
C IF NON-LINEARLY SPACED LEVELS ARE DESIRED, READ THEM NOW
C
          READ(5,220)(RLEV(I),I=1,ILEV)
220       FORMAT(8F10.5)
          LEVEL = ILEV
          GO TO 250
C
```

Figure 1 - continued

```
C SET UP POINTERS TO OVERLAY MAP SYMBOLS
C
230      DUM = VMIN
         DO 240 I=1,LEVEL
         DUM = DUM + VSUM
240      RLEV(I) = DUM
250      IF(LMAP .LE. 2) GO TO 902
         J = 0
         K = LEVEL - 1
         DO 260 I=1,K
260      J = J + I
C
         DO 270 I=1,LEVEL
          L = IL(J + I)
          ICHAR(I,1) = IP(L,1)
          ICHAR(I,2) = IP(L,2)
          ICHAR(I,3) = IP(L,3)
270      ICHAR(I,4) = IP(L,4)
C
C PRINT OVERLAID MAP
C
       DO 41 I=1,NTIME
       J=JJ(I)
          N = J + 1
         CALL BORDER(I,J,"1")
       DO 42 K=1,JROW
         READ(INPCO) (PTS(L),L=1,J)
       DO 48 L=1,4
       DO 48 M=1,130
       IPRT(L,M)=BLANK
   48  CONTINUE
C
       DO 43 L=1,J
       IF(PTS(L)) 43,43,44
   44  DO 46 M=1,LEVEL
       IF(PTS(L)-RLEV(M)) 47,47,46
   46  CONTINUE
       M=LEVEL
   47  IPRT(1,L)=ICHAR(M,1)
       IPRT(2,L)=ICHAR(M,2)
```

Figure 1 - continued

```
      IPRT(3,L)=ICHAR(M,3)
      IPRT(4,L)=ICHAR(M,4)
        IAREA(M) = IAREA(M) + 1
        ATOT = ATOT + 1.
   43 CONTINUE
C
C BORDER CHANGE FROM "I" AT INCREMENTS DIVISIBLE BY 5 AND 10
C
         M = K / 5
         IF(M * 5 .NE. K) GO TO 101
         IBRD = "+"
         M = M / 2
         IF(M * 10 .NE. K) GO TO 101
         M = MOD(M,10) + 1
         IBRD = IPRTCR(M)
101     IPRT(1,N) = IBRD
C
      WRITE(6,105) IBRD,(IPRT(1,M),M=1,N)
      WRITE(6,106) (IPRT(2,M),M=1,J)
      WRITE(6,106) (IPRT(3,M),M=1,J)
      WRITE(6,106) (IPRT(4,M),M=1,J)
105     FORMAT(1H ,132A1)
106     FORMAT(1H+,1X,131A1)
        IBRD = "I"
   42 CONTINUE
        CALL BORDER(I,J,BLANK)
   41 CONTINUE
      GO TO 49
C
C PRINT NON-OVERLAID MAP
C
 902  DO 941 I=1,NTIME
      J=JJ(I)
        N = J + 1
        CALL BORDER(I,J,"1")
      DO 942 K=1,JROW




        READ(INPCO)(PTS(L),L=1,J)
      DO 943 L=1,J
      IPRT(1,L)=BLANK
      IF(PTS(L)) 943,943,944
```

Figure 1 - continued

```
 944   DO 946 M=1,LEVEL
       IF(PTS(L)-RLEV(M))  947,947,946
 946   CONTINUE
       M=LEVEL
 947   IPRT(1,L)=ISYM(M)
       IAREA(M)=IAREA(M)+1
         ATOT = ATOT + 1
 943   CONTINUE
          M = K / 5
          IF(M * 5 .NE. K) GO TO 940
          IBRD = "+"
          M = M / 2
          IF(M * 10 .NE. K) GO TO 940
          M = MOD(M,10) + 1
          IBRD = IPRTCR(M)
940       IPRT(1,N) = IBRD
       WRITE(6,105) IBRD,(IPRT(1,M),M=1,N)
         IBRD = "I"
 942   CONTINUE
         CALL BORDER(I,J,BLANK)
941      CONTINUE
C
C READ HOWEVER MANY COMMENTS THERE ARE
C
   49 WRITE(6,91)
91       FORMAT(////)
92       READ(5,93)(STS(I),I=1,16)
93       FORMAT(16A5)
C
C    CHECK FOR END OF COMMENTS
C
         IF(STS(1) .EQ. 5HEND  ) GO TO 95
         WRITE(6,94)(STS(I),I=1,16)
94       FORMAT(1X,16A5)
         GO TO 92
C
C SET UP FOR BEGINNING OF SUMMARY
C
95       WRITE(6,300) VMIN,VMAX
300      FORMAT(////," ABOVE MAP HAS A MINIMUM OF:",F10.3,/
     1           ,17X,"MAXIMUM OF:",F10.3)
         IF(FLIP .GT. 0) WRITE(6,310)
310      FORMAT(" INCOMING MAP WAS INVERTED")
         IF(MOD(LMAP,2) .EQ. 0) GO TO 800
C
```

Figure 1 - continued

```
C ASQ IS SCALER FOR RECTANGULAR CELLS (.10 WIDE BY .125 TALL)
C DO 917 COMPUTES AREAS IN ACRES, SQ. MILES, HECTARES, AND PERCENT
C  (RESPECTIVELY) FOR AREA SUMMARY IF DESIRED
C
      ASQ=(SCALE*SCALE)/(144.*80.)
      DO 917 I=1,LEVEL
      AREA(1,I)=(IAREA(I)*ASQ)/43560.
      AREA(3,I)=AREA(1,I)/640.
      AREA(2,I)=(IAREA(I)*ASQ)/107600.
917   AREA(4,I)=(IAREA(I)/ATOT)*100.
C
C SET UP FOR MAX AND MIN PRINTING
C
800      L = LEVEL + 2
         DO 916 I=1,LEVEL
          K = L - I
          J = K - 1
916       RLEV(K) = RLEV(J)
         RLEV(1) = VMIN
         ML = -9
         IH = 0
C
C SET UP DATA LEVEL OUTPUT FOR BOTH TYPES OF MAP
C
810      ML = ML + 10
         IH = IH + 10
C
C IF IH AND ML ARE = OR > LEVEL, WE"RE DONE
C
         IF(IH .LE. LEVEL) GO TO 820
          IF(ML .GT. LEVEL) GO TO 999
          IH = LEVEL
820      IF(LMAP .LT. 3) GO TO 830
C
C OVER PRINT SETUP FOR PRINTING SYMBOLS FOR LEVELS (9 SYMBOLS,
C  1 BLANK)
C
      K1=0
      DO 520 I=1,LEVEL
      DO 521 J=1,9
```

Figure 1 - continued

```
      K1=K1+1
      IPRT(1,K1)=ICHAR(I,1)
      IPRT(2,K1)=ICHAR(I,2)
      IPRT(3,K1)=ICHAR(I,3)
      IPRT(4,K1)=ICHAR(I,4)
  521 CONTINUE
      K1=K1+1
      IPRT(1,K1)=BLANK
      IPRT(2,K1)=BLANK
      IPRT(3,K1)=BLANK
520   IPRT(4,K1) = BLANK
        GO TO 840
C
C NON OVER-PRINT SYMBOL SETUP
C
830     K1 = 0
        DO 923 I=ML,IH
         DO 924 J=1,9
          K1 = K1 + 1
924       IPRT(1,K1) = ISYM(I)
         K1 = K1 + 1
923      IPRT(1,K1) = BLANK
C
C PRINT MAX AND MIN
C
840     L = ML + 1
        M = IH + 1
        WRITE(6,912)
912     FORMAT(1HO," VALUE RANGE FOR EACH LEVEL")
        WRITE(6,913)(RLEV(I),I=ML,IH)
913     FORMAT(1HO,"MINIMUM",10F10.3)
        WRITE(6,914) (RLEV(I),I=L,M)
914     FORMAT(1HO,"MAXIMUM",10F10.3)



C
C SKIP AREA SUMMARY IF IT"S NOT DESIRED
C
        IF(MOD(LMAP,2) .EQ. 0) GO TO 850
      WRITE(6,915) (IAREA(I),I=ML,IH)
915     FORMAT(1HO, "CELLS  "   ,10I10)
      WRITE(6,918) (AREA(1,I),I=ML,IH)
```

Figure 1 - continued

```
918      FORMAT(1HO, "ACREAGE",10F10.2)
       WRITE(6,919) (AREA(2,I),I=ML,IH)
919      FORMAT(1HO, "HECTARS",10F10.2)
       WRITE(6,920)(AREA(3,I),I=ML,IH)
920      FORMAT(1HO, "SQ.MILES",F9.4,9F10.4)
       WRITE(6,921) (AREA(4,I),I=ML,IH)
921      FORMAT(1HO, "PERCENT",10F10.2)
C
C PRINT SYMBOLS FOR LEVELS
C
850      WRITE(6,860)
860      FORMAT(1HO," SYMBOLS FOR DATA LEVELS")
         DO 532 I=1,5
          WRITE(6,533) (IPRT(1,J),J=1,K1)
          WRITE(6,534) (IPRT(2,J),J=1,K1)
          WRITE(6,534) (IPRT(3,J),J=1,K1)
532       WRITE(6,534) (IPRT(4,J),J=1,K1)
533      FORMAT(1H ,8X,100A1)
534      FORMAT(1H+,8X,100A1)
C
C NOW GO AND SEE IF THERE ARE MORE THAN 10 LEVELS
C
         GO TO 810
C
C  FINISH IT OFF
C
999      DC = ATOT * ASQ
         CC = DC / 43560.
         CR = DC / 107600.
         DC = CC / 640.
         WRITE(6,1000) ATOT,CC,CR,DC
1000     FORMAT(//," TOTAL AREAS:",F7.0," CELLS",F10.2," ACRES",F10.2,
     1          " HECTARES",F10.2," SQUARE MILES")
         REWIND INPCO
         REWIND 12
         IF(IABS(IMP1 + IMP2) .EQ. 0 .AND. IMAP .EQ. 1)
     1                    GO TO 710
         STS(1) = .08
         STS(2) = .10
         WRITE(12) JROW,JCOL,STS(1),STS(2),VMIN,VMAX
         DO 700 I=1,NTIME
          J = JJ(I)
          DO 700 K=1,JROW
           READ(INPCO) (PTS(L),L=1,J)
           WRITE(12) (PTS(L),L=1,J)
```

Figure 1 - continued

```
700        CONTINUE
C
         WRITE(6,720)
720      FORMAT(" END COMPLEX RUN - NEW DATA FILE WRITTEN TO TAPE12")
C
 710     RETURN
         END
         SUBROUTINE READR
         COMMON INP(2),IMP(2),IFORM(16),PTS(774)



         EQUIVALENCE (INP1,INP(1)),(INP(2),INP2)
         DIMENSION NCOL(10)
         DATA NCOL/10*130/,ZERO/0.0/
C
C GO TO 80 IF DATA NOT IN SYMAP FORM
C
         IF(IABS(IMP1) .GT. 1) GO TO 80
         ISWT = 1
         IF(IMP(1).EQ.0) ISWT=2
C
C READ FORMAT FOR THE INITIAL MIN - MAX RECORD
C
15       READ(5,10) IFORM
10       FORMAT(16A5)
         INPT = INP(ISWT)
         READ(INPT,IFORM) IROW,ICOL,VMIN,VMAX
         INPCO1 = INP(ISWT) + 1
         WRITE(INPCO1) IROW,ICOL,ZERO,ZERO,VMIN,VMAX
C
C FORMAT FOR READING THE BODY OF THE DATA
C
         READ(5,10) IFORM
         NT = ICOL / 130
         IF(NT * 130 .LT. ICOL) NT = NT + 1
         NCOL(NT) = ICOL - (NT - 1) * 130
         IF(IMP(ISWT).EQ.-1) GO TO 30
C
C HERE WE"LL READ THE CODED DATA BASE (ASSUMED TO BE IN SYMAP FORM)
C NOTE THAT THE FORMAT INPUT IS SUPPOSED TO DO ANY JUMPING ABOUT
C  FOR PICKING UP THE CORRECT DATA POINTS IN THE DATA BASE
C
```

Figure 1 - continued

```
      DO 20 I=1, NT
       J = NCOL(I)
       DO 20 K=1,IROW
        READ(INPT,IFORM)(PTS(L),L=1,J)
20      WRITE(INPC01) (PTS(L),L=1,J)
C
25    INP(ISWT) = INPC01
      REWIND INPC01
      IF(ISWT .EQ. 2) RETURN
      IF(IMP(2).EQ.0) RETURN
      ISWT=2
      GO TO 15
C
C COME HERE FOR NON SYMAP TYPE DATA. IT"S NECESSARY TO HANDLE THIS
C  DATA DIFFERENTLY BECAUSE OF THE LIMITATIONS IN THE MAINLINE PROGRAM
C NOTE THAT THE CASE OF ICOL <= 130 IS A TRIVIAL CASE OF THE NON SYMAP
C  CASE, HANDLED BY THE SYMAP FORM CODE ABOVE
C
30    K = 0
      DO 40 I=1,NT
       K = K + 130
40     NCOL(I) = K
C
      DO 60 I=1,IROW
       READ(INPT,IFORM)(PTS(J),J=1,ICOL)
       WRITE(INPC01)(PTS(J),J=1,130)
       DO 50 J=2,NT
        M = NCOL(J-1) + 1
        N = NCOL(J)
50      WRITE(10) J,(PTS(K),K=M,N)




60      CONTINUE
C
C NOW READ DATA FROM THE TAPE WHERE IT WAS STORED (TAPE10) AND GET IT
C  TO THE DATA TAPE IN THE CORRECT ORDER
C
      NCOL(NT) = ICOL
      DO 70 I=2,NT
       J = NCOL(I) - (I-1) * 130
       REWIND 10
       DO 75 K=1,IROW
```

Figure 1 - continued

```
          READ(10) L,(PTS(M),M=1,130)
          IF(L .NE. I) GO TO 75
          WRITE(INPC01) (PTS(M),M=1,J)
75        CONTINUE
70       CONTINUE
        REWIND 10
        GO TO 25
C
C COME HERE FOR BOTH MAPS FROM THE SAME DATA BASE
C
80      INP1 = 9
        INP2 = 12
        READ(5,10) IFORM
        IF(IMP2 .LE. 0) GO TO 90
        INP1 = 12
        INP2 = 9
90      READ(8,IFORM) IROW,ICOL,(PTS(I),I=1,4)
        WRITE(INP1) IROW,ICOL,ZERO,ZERO,PTS(1),PTS(2)
        WRITE(INP2) IROW,ICOL,ZERO,ZERO,PTS(3),PTS(4)
        READ(5,10) IFORM
        NT = ICOL / 130
        IF(NT * 130 .LT. ICOL) NT = NT + 1
        IF(IMP1 .LT. 0) GO TO 120
        NCOL(NT) = ICOL - (NT -1) * 130
         DO 100 I=1,NT
         J = NCOL(I) * 2
         DO 110 K=1,IROW
          READ(8,IFORM)(PTS(L),L=1,J)
          WRITE(INP1)(PTS(L),L=1,J,2)
          WRITE(INP2)(PTS(L),L=2,J,2)
110       CONTINUE
100     CONTINUE
C
        REWIND INP1
        REWIND INP2
        RETURN
C
C HERE IF FILE IN NON SYMAP FORM AND BOTH MAPS FROM SAME DATA BASE
C
120     K = 0
        DO 130 I=1,NT
         NCOL(I) = K
130     K = K + 130
C
        K = ICOL * 2
```

Figure 1 - continued

```
      DO 140 I=1,IROW

       READ(8,IFORM)(PTS(J),J=1,K)
       WRITE(INP1)(PTS(J),J=1,260,2)
       WRITE(INP2)(PTS(J),J=2,260,2)
       DO 150 J=2,NT



        M = NCOL(I-1) * 2 + 1
        N = NCOL(I) * 2
        WRITE(10) J,(PTS(L),L=M,N,2)
        M = M + 1
150     WRITE(11) J,(PTS(L),L=M,N,2)
140   CONTINUE
C
      NCOL(NT) = IROW
      DO 160 I=2,NT
       REWIND 10
       REWIND 11
      K = NCOL(I) - NCOL(I - 1)
       DO 180 J=1,IROW
        READ(10) M,(PTS(L),L=1,130)
        IF(M .NE. I) GO TO 170
        WRITE(INP1) (PTS(L),L=1,K)
170     READ(11) M,(PTS(L),L=1,130)
        IF(M .NE. I) GO TO 180
        WRITE(INP2) (PTS(L),L=1,K)
180    CONTINUE
160   CONTINUE
C
      REWIND INP1
      REWIND INP2
      REWIND 10
      RETURN
      END
      SUBROUTINE BORDER(IST,IEND,IFST)
C
C THIS ROUTINE PUTS AN UPPER OR LOWER BORDER AROUND A MAP.
C THE BORDER WILL HAVE THE COUMNS NUMBERED ACCORDING TO:
C       IST     HAS THE NUMBER THE NUMBERING STARTS
C       IEND    WHERE THE NUMBERING ENDS
C       IFST    THE CHARACTER TO BE PRINTED IN THE FIRST PRINT LOCATION
C
```

Figure 1 - continued

```
      COMMON IDUM(4),ISYMB(10),IPRT(132)
      DATA IS,IDASH,IP/"*","-","+"/

      DO 10 I=1,132
10    IPRT(I) = IDASH

      DO 20 I=1,130,5
20    IPRT(I) = IP

      IST1 = MOD(IST *13 -13,10) + 1
      DO 30 I=1,IEND,10
       IPRT(I) = ISYMB(IST1)
30     IST1 = MOD(IST1,10) + 1

      IE = IEND + 2
      IPRT(1) = IS
      IPRT(IE) = IS
      WRITE(6,40) IFST,(IPRT(I),I=1,IE)
40    FORMAT(133A1)
      RETURN
      END

      SUBROUTINE PSORCA(NRUN)
      DIMENSION TITLE(16),MROW(20),MCOL(20),VALSR(260),ISEC(4)
      COMMON VAL1(260),VAL2(260),VAL3(260),JROW(500),
     1 JCOL(500),G(130),A(500),IROW,ICOL,W1,W2,W3,NCOL,NPT,




     1 RLENGTH,NROW,SUM,KO,NOB,KF,ISL,ANG(8),TANG(8)

C  CHECK WIDTH OF MATRIX
      REWIND 1
      READ (1) NROW,NCOL
      IF(NCOL.LE.130) GO TO 700
      CALL COPY (NROW,NCOL)

C  SET ANGLES
 700  ANG(1)=-2.24554
      ANG(2)=-1.57080
      ANG(3)=-0.89606
      ANG(4)=0.
      ANG(5)=0.89606
```

Figure 1 - continued

```
      ANG(6)=1.57080
      ANG(7)=2.24554
      ANG(8)=3.14159
      TANG(1)=-2.69357
      TANG(2)=-1.90817
      TANG(3)=-1.23343
      TANG(4)=-.44803
      TANG(5)=.44803
      TANG(6)=1.23343
      TANG(7)=1.90817
      TANG(8)=2.69357

C  PRINT HEADER
      PRINT 1
 1    FORMAT("1"///," ***PSORCA***"//," A CORRIDOR SELECTION PROGRAM"/)

C  READ AND PRINT TITLE
      READ 2,(TITLE(I),I=1,16)
 2    FORMAT(16A5)
      PRINT 2,(TITLE(I),I=1,16)

      DO 95 IRUN=1,NRUN

C  REPOSITION FILES AND RESET COUNTERS
      REWIND 2
      REWIND 1
      READ(1)
      NPT=1
      RLENGTH=0.

C  READ IN NUMBER OF POINT SPECIFICATIONS, DIRECTION-FINDING WEIGHTS,
C  AND INITIAL DIRECTION
      READ 5,NSPEC,DW1,DW2,DW3,KO,NOB,IMUL,ISL
 5    FORMAT(I10,3F10.0,4I10)
      IF(ISL.NE.0) NOB=1

C  READ IN SEARCH DISTANCES AND GUIDE
      MINDX=0
      MAXDX=0
      IF (IMUL.NE.0) READ 900,MINDX,MAXDX,DGUIDE
 900  FORMAT (2I10,F10.0)

C  SET DEFAULTS FOR WEIGHTS
      W1=2.*DW1
      W2=20.*DW2
```

Figure 1 - continued

```
      W3=3.*DW3
      GUIDE=.5*DGUIDE



      IF (KO.EQ.0) KO=6

C  READ IN POINT SPECIFICATIONS
      READ 201,(MROW(I),MCOL(I),I=1,NSPEC)
 201  FORMAT (2I10)
      IROW=MROW(1)
      ICOL=MCOL(1)

C  PRINT INPUT SPECIFICATIONS
      PRINT 6,IRUN,(TITLE(I),I=1,16),MROW(1),MCOL(1),MROW(NSPEC
     1 ),MCOL(NSPEC),DW1,DW2,DW3
 6    FORMAT ("1",///" **CORRIDOR ",I3,"**"//,16A5,///
     1 " START AT ROW",I5," AND COLUMN",I5,"."/
     1 " FINISH AT ROW",I5," AND COLUMN",I5,"."/
     2 " WEIGHTS=",3(1X,F8.2)//)
      IF(NOB.EQ.0) PRINT 8
 8    FORMAT(" NO BACKUP ON DESCENTS"/)
      IF(ISL.NE.0) PRINT 99
 99   FORMAT(" straight line corridor")
      IDX=MAXDX-MINDX
      IF (IDX.NE.0) PRINT 202,DGUIDE,MINDX,MAXDX
 202  FORMAT (" THIS IS A MULTISTAGE SEARCH"/" GUIDE=",F10.2/
     1 " MINDX=",I5," ; MAXDX=",I5)
      NPRT=NSPEC-2
      N1=NSPEC-1
      IF(NPRT.NE.0) PRINT 203,NPRT
 203  FORMAT(/" THERE ARE",I5," INTERMEDIATE POINTS SPECIFIED"/)
      IF (NPRT.EQ.0) GO TO 204
      PRINT 205
 205  FORMAT(13X,"ROW" 4X,"COLUMN")
      PRINT 206, (MROW(I),MCOL(I),I=2,N1)
 206  FORMAT(6X,2I10)

C  PRINT HEADING
 204  PRINT 7
 7    FORMAT(///" *CORRIDOR POINTS"//" POINT",
     1 7X,"ROW",4X,"COLUMN",5X,"VALUE"/)
```

Figure 1 - continued

```
C  READ IN INITIAL VALUES
        IF(IROW.EQ.2) GO TO 601
        IF (IROW.EQ.1) GO TO 602
        NCOUNT=IROW-2
        DO 11 I=1,NCOUNT
 11     READ (1)
        IF (NCOL.LE.130) GO TO 601
        DO 701 I=1,NCOUNT
 701    READ (2)
 601    CALL READVL (VAL1,NCOL)
 602    CALL READVL (VAL2,NCOL)
        CALL READVL (VAL3,NCOL)

C  RECORD FIRST CORRIDOR POINT
        JROW(NPT)=IROW
        JCOL(NPT)=ICOL
        A(NPT)="S"
        PRINT 207, NPT,IROW,ICOL,VAL2(ICOL)
        WRITE (60,208) IROW,ICOL
 208    FORMAT(2I10)
 207    FORMAT (I6,2I10,F10.2," (START)")
        SUM=VAL2(ICOL)



C  BEGIN CORRIDOR SELECTION PROCEDURE
        DO 250 IPT=1,N1
        IF (IDX.EQ.0) GO TO 240

C  TEST TO SEE IF SEARCH IS NEEDED
        M1=MROW(IPT+1)
        M2=MCOL(IPT+1)
 209    ITD=SQRT((M1-IROW)*(M1-IROW)+(M2-ICOL)*(M2-ICOL)*.64)
        IF(ITD.LT.MAXDX) GO TO 240
        Z1=M1-IROW
        Z2=M2-ICOL
        Z2=.8*Z2
        PHI=ATAN2(Z1,Z2)
        MX1=IROW+MINDX*SIN(PHI)
        MY1=ICOL+MINDX*COS(PHI)*1.25
        MX2=IROW+MAXDX*SIN(PHI)
        MY2=ICOL+MAXDX*COS(PHI)*1.25
```

Figure 1 - continued

```
C  SEARCH FOR NEXT POINT
      BESTSR=0.
      IF(Z1.EQ.0.) GO TO 505
      ISEC(1)=MX1-(Z2/Z1)*.8*(NCOL-MY1)
      ISEC(2)=MX2-(Z2/Z1)*.8*(NCOL-MY2)
      ISEC(3)=MX2+(Z2/Z1)*.8*(MY2)
      ISEC(4)=MX1+(Z2/Z1)*.8*(MY1)
      MINROW=NROW
      MAXROW=1
      DO 510 I=1,4
      IF(ISEC(I).LT.1) ISEC(I)=1
      IF(ISEC(I).GT.NROW) ISEC(I)=NROW
      IF(ISEC(I).GT.MAXROW) MAXROW=ISEC(I)
510   IF(ISEC(I).LT.MINROW) MINROW=ISEC(I)
      GO TO 502
505   MINROW=1
      MAXROW=NROW
502   REWIND 1
      IF (NCOL.GT.130) REWIND 2
      DO 500 I=1,MINROW
500   READ (1)
      IF(NCOL.LE.130) GO TO 501
      DO 704 I=1,MINROW
704   READ(2)
      BACKSPACE 2
501   DO 210 MR=MINROW,MAXROW
      CALL READVL (VALSR,NCOL)
      IF(Z2.EQ.0.) GO TO 507
      MINCOL=(Z1/Z2)*(MX1-MR)*1.25+MY1
      MAXCOL=(Z1/Z2)*(MX2-MR)*1.25+MY2
      GO TO 508
507   MINCOL=1
      MAXCOL=NCOL
508   IF(MINCOL.LE.MAXCOL)GO TO 503
      IDUM=MINCOL
      MINCOL=MAXCOL
      MAXCOL=IDUM
503   IF(MINCOL.LT.1) MINCOL=1
      IF(MAXCOL.GT.NCOL) MAXCOL=NCOL
      DO 210 MC=MINCOL,MAXCOL
      Z3=MC-ICOL
      Z3=.8*Z3
      Z4=MR-IROW
```

Figure 1 - continued

```
      ALPHA=1.571-PHI-ATAN2(Z3,Z4)
      DIF =SQRT(Z4*Z4+Z3*Z3)*SIN(ALPHA)
      TESTSR=VALSR(MC)-(ABS(DIF))*GUIDE
      IF (TESTSR.LT.BESTSR) GO TO 210
      BESTSR=TESTSR
      NEXTROW=MR
      NEXTCOL=MC
 210  CONTINUE

C  REPOSITION MATRIX
      REWIND 1
      DO 220 I=1,IROW
 220  READ (1)
      READ (1)
      READ(1)
      IF (NCOL.LE.130) GO TO 751
      REWIND 2
      DO 750 I=1,IROW
 750  READ (2)
      READ (2)
 751  CONTINUE

C  SELECT NEXT CORRIDOR SEGMENT
      CALL SELECT(NEXTCOL,NEXTROW)
      JROW(NPT)=IROW
      JCOL(NPT)=ICOL
      A(NPT)="+"
      PRINT 221, NPT,IROW,ICOL,VAL2(ICOL)
      WRITE (60,208) IROW,ICOL
 221  FORMAT(I6,2I10,F10.2," (END STAGE)")
      SUM=SUM+VAL2(ICOL)
      GO TO 209

C  SELECT SEGMENT LEADING TO NEXT SPECIFIED POINT
 240  CALL SELECT(MCOL(IPT+1),MROW(IPT+1))
      JROW(NPT)=IROW
      JCOL(NPT)=ICOL
      A(NPT)="S"
      PRINT 251, NPT,IROW,ICOL,VAL2(ICOL)
      WRITE (60,208) IROW,ICOL
 251  FORMAT (I6,2I10,F10.2," (SPECIFIED)")
 250  SUM=SUM+VAL2(ICOL)

C  PRINT CORRIDOR LENGTH AND AVERAGE VALUE
      PRINT 80,RLENGTH
```

Figure 1 - continued

```
 80      FORMAT (///" CORRIDOR IS",F8.2," CELL UNITS LONG",///)
         PT=NPT
         AVER=SUM/PT
         PRINT 96, AVER
 96      FORMAT(//" AVERAGE VALUE OF CORRIDOR IS",F10.2)

C  PRINT MAP
         WRITE (60,97)
 97      FORMAT("99999")
         PRINT 85, IRUN,(TITLE(I),I=1,16)
 85      FORMAT("1",///" **CORRIDOR ",I3,"***"//16A5,///,
     1   " *CORRIDOR MAP"//)
         DO 71 J=1,NROW
         DO 69 J1=1,130
 69      G(J1)=" "
         DO 70 I=1,NPT




         IF (JROW(I).NE.J) GO TO 70
         ICOLG=JCOL(I)
         IF(ICOLG.GT.130) ICOLG=ICOLG-130
         G(ICOLG)=A(I)
 70      CONTINUE
 71      PRINT 72, (G(I1),I1=1,130)
 72      FORMAT (2X,130A1)
 95      CONTINUE


C  PRINT CONCLUDING REMARKS
         PRINT 100
 100     FORMAT (////" END OF RUN")
         RETURN
         END


         SUBROUTINE SELECT(NEXTCOL,NEXTROW)
         COMMON VAL1(260),VAL2(260),VAL3(260),JROW(500),JCOL(500),
      1  G(130),A(500),IROW,ICOL,W1,W2,W3,NCOL,NPT,RLENGTH,
      1  NROW,SUM,KO,NOB,KF,ISL,ANG(8),TANG(8)
         DIMENSION VALT(8),ICOLT(8),IROWT(8),CANG(8)

C  FIND DISTANCE AND DIRECTION TO NEXT POINT
```

Figure 1 - continued

```
      DISTNEX =SQRT((IROW-NEXTROW)*(IROW-NEXTROW)
     1  +(ICOL-NEXTCOL)*(ICOL-NEXTCOL)*.64)
      CLENGTH=RLENGTH
      PHI=ATAN2(FLOAT(NEXTROW-IROW),(NEXTCOL-ICOL)*.8)
      SMALL=3.14159
      DO 500 I=1,8
      TEST=PHI-TANG(I)
 500  IF(ABS(TEST).LE.SMALL) SMALL=TEST
      DO 501 I=1,8
 501  CANG(I)=ANG(I)+SMALL
      IF(CANG(8).GT.3.14159) CANG(8)=CANG(8)-6.28318

C  SET TEST POINT COORDINATES AND VALUES
 15   PATH =RLENGTH-CLENGTH
      VALT(1)=VAL1(ICOL-1)
      ICOLT(1)=ICOL-1
      IROWT(1)=IROW-1
      VALT(2)=VAL1(ICOL)
      ICOLT(2)=ICOL
      IROWT(2)=IROW-1
      VALT(3)=VAL1(ICOL+1)
      ICOLT(3)=ICOL+1
      IROWT(3)=IROW-1
      VALT(4)=VAL2(ICOL+1)
      ICOLT(4)=ICOL+1
      IROWT(4)=IROW
      VALT(5)=VAL3(ICOL+1)
      ICOLT(5)=ICOL+1
      IROWT(5)=IROW+1
      VALT(6)=VAL3(ICOL)
      ICOLT(6)=ICOL
      IROWT(6)=IROW+1
      VALT(7)=VAL3(ICOL-1)
      ICOLT(7)=ICOL-1
      IROWT(7)=IROW+1
      VALT(8)=VAL2(ICOL-1)
      ICOLT(8)=ICOL-1



      IROWT(8)=IROW
      DIST=SQRT((NEXTROW-IROW)*(NEXTROW-IROW)+
     1  (NEXTCOL-ICOL)*(NEXTCOL-ICOL)*.64)
      THETA=ATAN2(FLOAT(NEXTROW-IROW),(NEXTCOL-ICOL)*.8)
```

Figure 1 - continued

```
C  CHOOSE NEXT CORRIDOR POINT
        BEST=0.
        DO 20 I=1,3
        IT=KO-2+I
        IF(IT.LT.1) IT=IT+8
        IF(IT.GT.8) IT=IT-8
        DIFF=ABS(THETA-CANG(IT))
        IF(DIFF.GT.3.14159) DIFF=6.28318-DIFF
        DELTA=(3.14159-DIFF)/3.14159
        ADJ=DELTA*(W1*PATH/DISTNEX+W2/DIST+W3)
        IF(VALT(IT).LT.0.) VALT(IT)=0.
        Q=VALT(IT)+ADJ
        IF(ISL.NE.0) Q=DELTA
        IF (Q.LT.BEST) GO TO 20
        DO 19 J=1,NPT
        IF(JCOL(J).EQ.ICOLT(IT).AND.JROW(J).EQ.IROWT(IT))GO TO 20
 19     CONTINUE
        K=IT
        BEST=Q
 20     CONTINUE
        IF (KF.NE.1.OR.NOB.NE.0) KO=K
        KF=0
        IF(VALT(K).LT.VAL2(ICOL)) KF=1

C  RECORD CORRIDOR LENGTH
        IF(K.EQ.2.OR.K.EQ.6) DLENGTH=1.
        IF(K.EQ.4.OR.K.EQ.8) DLENGTH=.8
        IF(K.EQ.7.OR.K.EQ.5.OR.K.EQ.3.OR.K.EQ.1) DLENGTH=1.281
        RLENGTH = RLENGTH+DLENGTH

C  RESET CURRENT COLUMN
        ICOL=ICOLT(K)

C  TEST TO SEE IF NEW ROW IS TO BE READ
        IF (K.EQ.4.OR.K.EQ.8) GO TO 43
        IF (K.GE.5) GO TO 920

C  READ NEW ROW
        DO 921 I=1,NCOL
        VAL3(I)=VAL2(I)
 921    VAL2(I)=VAL1(I)
        JF=4
        IF(IROW.EQ.NROW) JF=3
        IROW=IROW-1
```

Figure 1 - continued

```
      IF(IROW.EQ.1) GO TO 922
      DO 923 J=1,JF
923   BACKSPACE 1
      IF(NCOL.LE.130) GO TO 925
      DO 924 J=1,jf
924   BACKSPACE 2
925   CALL READVL(VAL1,NCOL)
      READ(1)
      READ(1)
      IF (NCOL.LE.130) GO TO 926
      READ(2)
      READ(2)



926   GO TO 43
922   BACKSPACE 1
      IF(NCOL.GT.130) BACKSPACE 2
      DO 927 I=1,NCOL
927   VAL1(I)=0
      GO TO 43
920   DO 40 I=1,NCOL
      VAL1(I)=VAL2(I)
40    VAL2(I)=VAL3(I)
      IROW=IROW+1
      IF(IROW.EQ.NROW) GO TO 41
      CALL READVL (VAL3,NCOL)
      GO TO 43
41    DO 42 I=1,NCOL
42    VAL3(I)=0
43    CONTINUE
      IF (IROW.EQ.NEXTROW.AND.ICOL.EQ.NEXTCOL) GO TO 60

C  RECORD CORRIDOR POINT COORDINATES AND VALUE
      NPT=NPT+1
      SUM=SUM+VALT(K)
      PRINT 46, NPT,IROW,ICOL,VALT(K)
      WRITE(60,1) IROW,ICOL
1     FORMAT (2I10)
46    FORMAT(I6,2I10,F10.2)
      JROW(NPT)=IROW
      JCOL(NPT)=ICOL
```

Figure 1 - continued

```
      A(NPT)="*"
      GO TO 15
60    NPT=NPT+1
      RETURN
      END


      SUBROUTINE READVL (VAL,NCOL)
      DIMENSION VAL(260)

      IF (NCOL.LE.130) GO TO 10
      READ (1) (VAL(I),I=1,130)
      READ (2) (VAL(I),I=131,NCOL)
      GO TO 15
10    READ (1) (VAL(I),I=1,NCOL)
15    RETURN
      END


      SUBROUTINE COPY (NROW,NCOL)
      DIMENSION VAL(260)

      DO 5 I=1,NROW
5     READ (1)
      DO 10 J=1,NROW
      READ (1) (VAL(I),I=131,NCOL)
10    WRITE (2) (VAL(I),I=131,NCOL)
      RETURN
      END
```

Figure 1 - continued

```
    2    2
104                2.          1.        11  34000.
 WC ECON SURFACE
    (INVERTED)
END
 CORRIDOR THROUGH WC ECON SURFACE
          2       500.       500.      500.          8
        156         33
          9         58
          2       500.       500.      500.          4
          9         58
        156         33
```

Figure 1 - continued